



# **BEA WebLogic Java Adapter for Mainframe™**

## **Configuration and Administration Guide**

# Copyright

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

## Restricted Rights Legend

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

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

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

## Trademarks or Service Marks

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

All other trademarks are the property of their respective companies.

## BEA WebLogic Java Adapter for Mainframe Configuration and Administration Guide

Part Number	Date	Software Version
N/A	August 2002	5.1

---

# Contents

## **1. Introduction to WebLogic JAM Configuration and Administration**

WebLogic JAM Architecture .....	1-2
The Communication Resource Manager .....	1-3
The WebLogic JAM Gateway .....	1-3
WebLogic JAM Configuration .....	1-4
Configuring WebLogic JAM Connectivity .....	1-4
Integrating Applications With WebLogic JAM .....	1-5
Advanced WebLogic JAM Configuration Consideration .....	1-6
Using WebLogic JAM in the WebLogic Administration Console .....	1-6
WebLogic JAM Administration .....	1-8
Administration Via the WebLogic Administration Console .....	1-9
CRM Administration from the Command Line .....	1-9
Transaction Administration .....	1-9
Roadmap for Configuration and Administration with WebLogic JAM .....	1-10

## **2. Configuring WebLogic JAM Connectivity**

Understanding WebLogic JAM Connectivity .....	2-2
Getting Started with WebLogic JAM Connectivity .....	2-3
Step 1: Deploy WebLogic JAM in the WebLogic Server Environment .....	2-4
Adding the jam.jar File to the WebLogic Server CLASSPATH .....	2-5
Deploying the jam.ear File .....	2-6
Defining the WebLogic JAM Startup Class .....	2-8
Step 2: Define Where the CRM Will Run .....	2-10
Step 3: Create a Logical Unit for the CRM .....	2-11
Parameters for Establishing the CRM Logical Unit .....	2-12
WebLogic Administration Console Parameter for the CRM Logical Unit .....	

---

2-13	
Step 4: Connect the CRM to Back-End Systems on the Mainframe.....	2-14
Determining Connection Characteristics.....	2-14
Logmode Name.....	2-15
Sessions.....	2-15
Security Level.....	2-16
Connecting to a CICS Region.....	2-17
Parameters and Steps for Connecting to a CICS Region.....	2-17
WebLogic Administration Console Parameters for CICS Region	
Connectivity.....	2-21
Connecting to an IMS Region.....	2-23
Parameters and Steps for Connecting to an IMS Region.....	2-24
WebLogic Administration Console Parameters for IMS Region	
Connectivity.....	2-27
Connecting to a Batch Region.....	2-30
Parameters and Steps for Connecting to a Batch Region.....	2-30
WebLogic Administration Console Parameters for Batch Region	
Connectivity.....	2-32
Step 5: Enter Connectivity Information into WebLogic Administration Console .	
2-35	
Step 5.1: Create Region Definitions.....	2-35
Step 5.2: Create CRM Definitions.....	2-41
Step 5.3: Create CRM Link Definitions.....	2-46
Step 6: Define a WebLogic JAM Gateway.....	2-51
Step 7: Verify Your WebLogic JAM Connectivity Configuration.....	2-56

### **3. Integrating Applications With WebLogic JAM**

Understanding Application Integration with WebLogic JAM.....	3-1
Application Integration Considerations.....	3-2
Data Translation.....	3-2
Access to Mainframe and WebLogic Applications.....	3-2
Exposing Mainframe Applications to J2EE Clients.....	3-3
Exposing a CICS Distributed Program Link as a DPL Service.....	3-4
Steps for Exposing a DPL Program.....	3-4
WebLogic Administration Console Parameters for DPL Services.....	3-5
Exposing a Standard IMS Transaction Program as an APPC Service.....	3-11

---

Steps for Exposing a Standard IMS Transaction Program .....	3-11
WebLogic Administration Console Parameters for an IMS Transaction Program Exposed as an APPC Service .....	3-12
Exposing an APPC/MVS Transaction Program as an APPC Service .....	3-18
Steps for Exposing an APPC/MVS Transaction Program as an APPC Service .....	3-18
WebLogic Administration Console Parameters for an APPC Transaction Program Exposed as an APPC Service .....	3-19
Exposing J2EE Applications to Mainframe Clients .....	3-25
Configuring Mainframe Client Applications .....	3-25
CICS Distributed Program Link Clients .....	3-26
Standard IMS Programs .....	3-28
APPC Programs that Run in CICS, IMS or as a Batch Job .....	3-28
Exposing Enterprise Java Beans to the Mainframe .....	3-30
WebLogic Administration Console Parameters for Enterprise Java Beans Exposed to the Mainframe .....	3-30
Steps for Creating an Export Definition for an EJB .....	3-32
Exposing JMS Events to the Mainframe .....	3-35
WebLogic Administration Console Parameters for JMS Events Exposed to the Mainframe .....	3-35
Steps for Creating a JMS Event Definition .....	3-38

## 4. Scaling WebLogic JAM Configurations

Connecting to Multiple Regions .....	4-2
Connecting Multiple Gateways to a Single CRM .....	4-2
Advantages of Multi-Gateway Support .....	4-4
Disadvantages of Multi-Gateway Configuration .....	4-4
Multi-Gateway Connection Issues .....	4-4
First WebLogic JAM Gateway Connection .....	4-5
Subsequent WebLogic JAM Gateway Connections .....	4-5
Multi-Gateway Disconnect and Shutdown Issues .....	4-5
Gateway Disconnect .....	4-6
Shutdown Processing (All Gateways Except the Last) .....	4-6
Shutdown Processing (Last Gateway) .....	4-6
Understanding Mainframe to WebLogic Load Balancing in a Multi-Gateway CRM .....	4-7

---

Support for a Clustered Environment .....	4-7
Clustered Gateways, Single CRM .....	4-8
Clustered Gateways, Multiple CRMs .....	4-9
Load Balancing Support with WebLogic Server .....	4-10
Failover Support with WebLogic Server .....	4-11
Failover for WebLogic to Mainframe Services .....	4-11
Failover for Mainframe to WebLogic Services .....	4-11
Support for Distributed Transactions .....	4-12
Transactional Affinity .....	4-12

## 5. Modifying Your Configuration

Modifying Your Connectivity Configuration .....	5-1
Modifying Region Definitions .....	5-2
Listing Region Definitions .....	5-2
Modifying the Logical Unit Name in a Region .....	5-4
Deleting a Region Definition .....	5-7
Modifying CRM Definitions .....	5-9
Listing CRM Definitions .....	5-10
Editing a CRM Definition .....	5-12
Deleting a CRM Definition .....	5-14
Modifying CRM Link Definitions .....	5-17
Listing CRM Link Definitions .....	5-18
Editing an Existing CRM Link .....	5-21
Deleting a CRM Link .....	5-25
Modifying Gateway Definitions .....	5-28
Listing Gateway Definitions .....	5-29
Editing a Gateway Definition .....	5-31
Deleting a Gateway Definition .....	5-34
Modifying Your Application Integration Configuration .....	5-36
Modifying a DPL Service Definition .....	5-37
Editing a DPL Service Definition .....	5-38
Deleting a DPL Service Definition .....	5-41
Modifying an APPC Service Definition .....	5-43
Editing an APPC Service Definition .....	5-44
Deleting an APPC Service Definition .....	5-47

---

Modifying an Exported EJB Definition .....	5-49
Editing an Exported EJB Definition .....	5-50
Deleting an Exported EJB Definition .....	5-53
Modifying a JMS Event Definition .....	5-55
Editing a JMS Event .....	5-56
Deleting a JMS Event .....	5-59
Modifying a WebLogic Integration Event Definition .....	5-61
Editing a WebLogic Integration Event .....	5-62
Deleting a WebLogic Integration Event .....	5-64

## **6. Administration and Diagnostics Using the WebLogic Administration Console**

Monitoring WebLogic JAM CRM Status .....	6-2
Modifying CRM Trace Level Settings .....	6-6
Setting APPC API Tracing in the WebLogic Administration Console .....	6-13
Monitoring CRM Links .....	6-17
Starting and Stopping CRM Links .....	6-23
Gateway Administration and Diagnostics .....	6-28
Listing Gateways .....	6-29
Monitoring a Gateway .....	6-31
Modifying Trace Level Settings for the Gateway .....	6-35
Starting and Stopping a Gateway .....	6-41
Enabling a Service .....	6-46
Enabling an Exported WebLogic Application .....	6-49

## **7. CRM Administration**

Starting the CRM .....	7-1
Setting CRM Trace Level at CRM Startup .....	7-3
Setting APPC API Tracing at Startup .....	7-4
Starting the CRM in an MVS Environment .....	7-4
Starting CRM in z/OS or OS/390 Unix Environment .....	7-7
Stopping the CRM .....	7-8
Stopping the CRM in an MVS Environment .....	7-9
Stopping the CRM in a z/OS or OS/390 Unix Environment .....	7-11
Activating and De-Activating CRM Links .....	7-11

---

Activating CRM Links .....	7-12
Activating CRM Links in an MVS Environment.....	7-13
Activating CRM Links from a z/OS or OS/390Unix Environment ..	7-15
De-Activating CRM Links .....	7-15
De-Activating a CRM Link in an MVS Environment .....	7-16
De-Activating a CRM Link in a z/OS or OS/390 Unix Environment .....	7-18
CRM Console Commands .....	7-19

## **8. Administering Transactions**

Configuring Logical Units with Transactional Support .....	8-2
Configure the CRM Logical Unit.....	8-2
Configure the Back-end Environment.....	8-3
Transaction Log Files .....	8-3
JTA Log Files .....	8-4
WebLogic JAM CRM Log Files .....	8-4
Restart Log File.....	8-4
Blob Log File .....	8-4
Viewing CRM Log Files .....	8-5
Mainframe Application Log.....	8-8
Restarting Links (cold/warm) .....	8-8
Cold Start.....	8-9
Starting a Link with a Cold Start.....	8-9
Warm Start.....	8-10
Recovery .....	8-10
Initiating Recovery .....	8-11

## **9. Preparing Your System for Planned Outages**

Shutting Down a Remote Region .....	9-2
Shutting Down a Mainframe .....	9-6
Shutting Down an Instance of WebLogic Server .....	9-6
Unplanned Outages.....	9-6
Application Region Terminates.....	9-7
z/OS or OS/390 Region Terminates .....	9-7



---

## A. Mainframe Connectivity Worksheet

CRM Definitions .....	A-1
Regions.....	A-2
CICS Regions.....	A-3
IMS Regions .....	A-3
Batch Regions .....	A-3
CRM Links.....	A-4
CICS.....	A-4
IMS.....	A-5
Batch .....	A-6



# **1 Introduction to WebLogic JAM Configuration and Administration**

BEA WebLogic Java Adapter for Mainframe (JAM) provides a gateway between multiple application servers (WebLogic Server, CICS, and IMS), allowing applications to collaborate while running in different environments and on different hosts. WebLogic JAM allows applications running on WebLogic Server and applications running on your CICS, IMS, or Batch systems to invoke each other. To enable this integration, you must configure WebLogic JAM to connect to your back-end systems.

The WebLogic Administration Console provides an easy-to-use interface for configuring and administering WebLogic JAM. After WebLogic JAM is installed, additional objects are available in the WebLogic Administration Console that you can use to configure and administer WebLogic JAM.

WebLogic JAM configuration involves working closely with system administration personnel to obtain network resources such as network addresses, logical units, and other back-end system communication information. Once network connectivity is established, you can use the WebLogic Administration Console to input additional configuration information for each application that is to be integrated. This task is often performed in conjunction with a CICS or IMS administrator / developer who is familiar with the applications that are to be integrated.

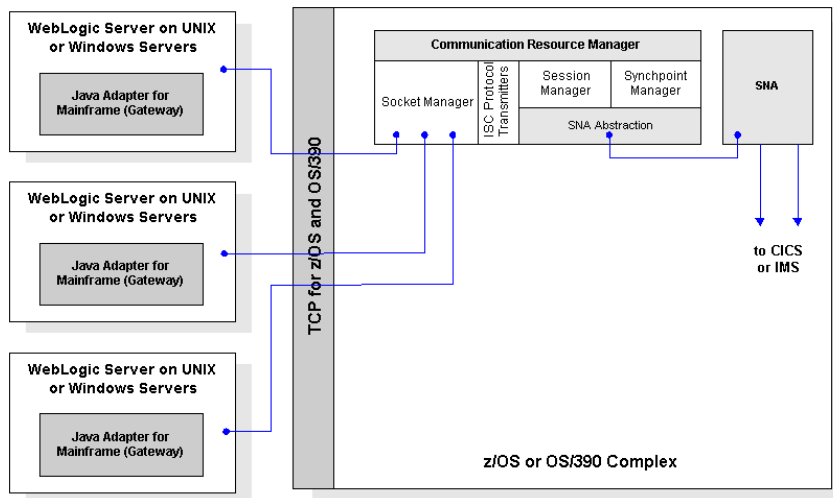
This section provides information on the following subjects:

- [WebLogic JAM Architecture](#)
- [WebLogic JAM Configuration](#)
- [WebLogic JAM Administration](#)
- [Roadmap for Configuration and Administration with WebLogic JAM](#)

## WebLogic JAM Architecture

WebLogic JAM allows applications running on BEA WebLogic Server and mainframe systems to interact using two main components: the WebLogic JAM Gateway and the Communications Resource Manager (CRM). These two WebLogic JAM components work together to allow applications to communicate even though they are running in disparate environments. Figure 1-1 provides an overview of WebLogic JAM components and connectivity.

**Figure 1-1 WebLogic JAM Environment**



## The Communication Resource Manager

The CRM is a component of WebLogic JAM that acts as an intermediary between the WebLogic JAM Gateway and mainframe systems, such as CICS or IMS.

Communications between the CRM and the WebLogic JAM Gateway use the TCP/IP protocol, while communications between the CRM and mainframe systems use APPC protocol.

The CRM is responsible for establishing and monitoring communications links that allow programs to communicate using different application protocols. The CRM also provides support for the coordination of transactional resources (such as databases) between systems – using the two-phase commit protocol.

For detailed information and instructions for configuring WebLogic JAM connectivity, including the CRM, see [“Configuring WebLogic JAM Connectivity.”](#)

## The WebLogic JAM Gateway

The WebLogic JAM Gateway is a component of WebLogic JAM that runs in the WebLogic environment and communicates with the CRM using TCP/IP protocol. The WebLogic JAM Gateway routes requests and responses between Java applications and mainframe systems, such as CICS and IMS. The WebLogic JAM Gateway also forwards configuration information to the CRM at start-up. Each WebLogic Server instance that needs to access back-end applications should have a WebLogic JAM Gateway defined.

In larger configurations, multiple instances of WebLogic Server may require connectivity to a variety of mainframe systems. To support this scenario, WebLogic JAM provides a multi-gateway CRM feature that allows a single instance of the CRM to service multiple instances of the WebLogic JAM Gateway. As a result, WebLogic Servers can share a single instance of the CRM as a common access point to mainframe systems.

For detailed information about configuring the Gateway, see [“Step 6: Define a WebLogic JAM Gateway.”](#) For detailed information about WebLogic JAM support for the multi-Gateway environment, see [“Connecting Multiple Gateways to a Single CRM.”](#)

# WebLogic JAM Configuration

WebLogic JAM configurations are created, edited, and updated using the WebLogic Administration Console. The *BEA WebLogic Java Adapter for Mainframe Installation Guide* provides instructions for installing and deploying WebLogic JAM so that you can access it from the WebLogic Administration Console.

WebLogic JAM configuration information is persisted on the administration node of the WebLogic domain. WebLogic JAM configuration is stored in Extensible Markup Language (XML) in a single file `jamconfig.xml`. The `jamconfig.xml` should reside in the domain-specific directory where WebLogic JAM runs.

Similar to WebLogic Server configuration, WebLogic JAM configuration is administered per WebLogic domain. A WebLogic domain is a self-contained administration unit that can include multiple WebLogic Servers and clusters. This domain is configured and administered as a group. For more information on WebLogic domains see Overview of WebLogic Server Management at <http://edocs.bea.com/wls/docs70/adminguide/overview.html>.

WebLogic JAM configuration information is domain-wide and shared among instances of WebLogic JAM that run in the domain. Because configuration information is shared throughout the domain, WebLogic JAM can be configured to support clustering, load-balancing, and failover.

There are two primary considerations when configuring WebLogic JAM using the WebLogic Administration Console:

- [Configuring WebLogic JAM Connectivity](#)
- [Integrating Applications With WebLogic JAM](#)

## Configuring WebLogic JAM Connectivity

WebLogic JAM uses two distributed software components to connect to your back-end systems: the WebLogic JAM Gateway and the Communications Resource Manager (CRM). In order for WebLogic JAM to work with your back-end systems, you will establish connectivity between the mainframe, the CRM and the Gateway.

Configuring the WebLogic JAM connectivity to the CRM requires several primary steps:

1. Decide where you want to install and run the CRM. A system administrator will probably perform the installation.
2. The system administrator adds a VTAM definition (a logical unit) to identify the CRM to the SNA network (VTAM).
3. The system administrator defines the back-end system (IMS or CICS) to VTAM to identify the back-end logical unit to the SNA network. Steps for this process vary depending on the type of back-end system.
4. The system administrator configures the back-end system to recognize the logical unit created in step 2 and/or 3. Steps for this process vary depending on the type of back-end system.
5. You use the WebLogic Administration Console to enter the connection parameters that were established in steps 1-4. Entering these parameters into the WebLogic Administration Console configures the CRM to recognize its logical unit and configure the partner logical unit to which it will connect.

For detailed information and instructions on configuring WebLogic JAM connectivity, see [“Configuring WebLogic JAM Connectivity.”](#)

## Integrating Applications With WebLogic JAM

WebLogic JAM allows applications running in the mainframe environment and the WebLogic environment to invoke each other. You can create a configuration entry via the WebLogic Administration Console to expose an application to the opposite environment. For information about configuring WebLogic JAM to work with these various types of applications, see [“Integrating Applications With WebLogic JAM.”](#)

# Advanced WebLogic JAM Configuration Consideration

WebLogic JAM can be configured to scale as your mainframe integration needs grow, and you can configure WebLogic JAM with redundant components to provide failover capabilities. WebLogic JAM also offers several features that allow you to add capacity as your needs grow and to add redundancy to provide high availability. These features include:

- WebLogic JAM support for a multiple Gateway CRM
- Support for a clustered environment

Detailed information about this support is available in [“Scaling WebLogic JAM Configurations.”](#)

## Using WebLogic JAM in the WebLogic Administration Console

The WebLogic Administration Console is a Web application that allows you to access the management resources in the Administration Server. Use the following steps to access WebLogic JAM in the WebLogic Administration Console.

**Note:** You must deploy the `jam.ear` file in your WebLogic JAM domain configuration in order to access WebLogic JAM from the WebLogic Administration Console. For information on how to deploy the `jam.ear` file, refer to [“Deploying the jam.ear File.”](#)

1. From your browser, open the WebLogic Administration Console using the following address:

```
http://hostname:7001/console
```

In this address, the following definitions apply:

`hostname` is the address of the machine where WebLogic Server is running.

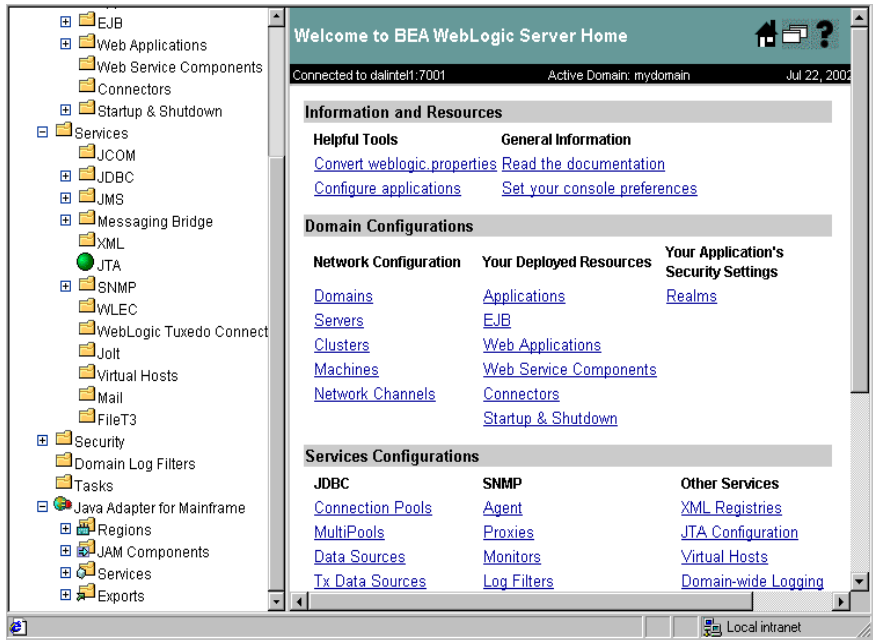
`7001` is the port for WebLogic Server that has been configured for the `examples` domain.



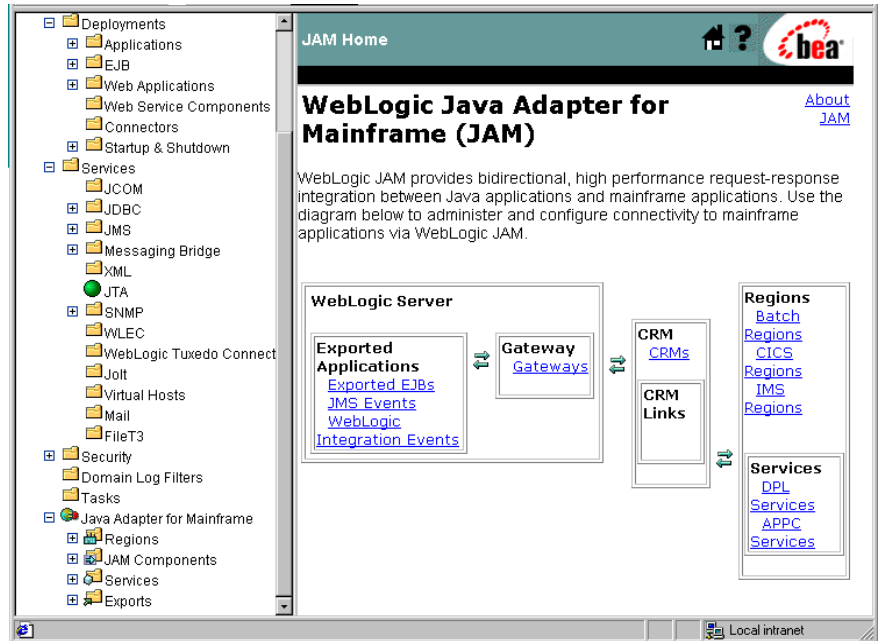
2. When prompted, supply your username and password that allows you access to the WebLogic Administration Console.

Refer to the installation information in the BEA WebLogic Server documentation for more information.

The WebLogic Administration Console displays.



3. Click **Java Adapter for Mainframe** in the left navigation bar. The WebLogic JAM home page displays.



## WebLogic JAM Administration

In addition to configuration of connectivity and mainframe/Java application integration, WebLogic JAM provides support for the following administration features:

- [Administration Via the WebLogic Administration Console](#)
- [CRM Administration from the Command Line](#)
- [Transaction Administration](#)

## Administration Via the WebLogic Administration Console

As with configuration, WebLogic JAM provides administrative support for the CRM and the Gateway via the WebLogic Administration Console. The WebLogic Administration Console provides you with the ability to quickly and easily monitor CRM and CRM link status, start and stop a CRM link, start and stop a Gateway, modify trace levels, and enable services. For detailed information about these WebLogic JAM administrative tasks, see [“Administration and Diagnostics Using the WebLogic Administration Console.”](#)

## CRM Administration from the Command Line

With WebLogic JAM, you run the CRM in an z/OS or OS/390 Unix environment or an MVS environment. Some of the administration tasks for these environments are performed from a command line or using JCL.

These tasks include starting and stopping the CRM, activating and de-activating CRM links, and various other CRM tasks. For detailed information about CRM Administration, see [“CRM Administration.”](#)

## Transaction Administration

WebLogic Server provides J2EE transactional support using Java Transactional API (JTA). WebLogic Server is the transaction manager. The WebLogic JAM Gateway will register with the transaction manager, and become part of a transaction when a request is made to a remote service. The remote service is part of a transaction under a different transaction manager, and the request becomes part of a distributed transaction. WebLogic JAM becomes a communication resource manager, and provides two-phase commit protocol using SNA to the communication resource manager under the foreign transaction manager. The foreign transaction manager is one of the following:

- CICS or CICS using RRS
- IMS using APPC/MVS and RRS
- Batch processes using APPC/MVS and RRS

For information about administering transactions and transaction log files, see [“Administering Transactions.”](#)

# Roadmap for Configuration and Administration with WebLogic JAM

The steps outlined in the following table provide you with a high-level guideline to all of the tasks and processes that you must perform to configure WebLogic JAM connectivity, configure WebLogic JAM for integrating applications, and administering WebLogic JAM. Think of these steps as a roadmap to guide you through the process and to point you to the resources available to help you.

To...	Refer To...
1. Determine where to run the CRM.	<a href="#">“Configuring WebLogic JAM Connectivity”</a>
2. Determine your application environments.	<a href="#">“Configuring WebLogic JAM Connectivity”</a>
3. Define and configure WebLogic and mainframe connectivity.	<a href="#">“Configuring WebLogic JAM Connectivity”</a>
4. Define and configure services to expose applications.	<a href="#">“Integrating Applications With WebLogic JAM”</a>
5. Scale WebLogic JAM configurations for multiple regions, multi-Gateway, or clustered environments.	<a href="#">“Scaling WebLogic JAM Configurations”</a>
6. Administer WebLogic JAM.	<a href="#">“Modifying Your Configuration”</a> , <a href="#">“Administration and Diagnostics Using the WebLogic Administration Console”</a> , and <a href="#">“CRM Administration”</a>

# 2 Configuring WebLogic JAM Connectivity

The WebLogic Administration Console provides you with the tools you need to configure connectivity between your mainframe and WebLogic Server.

This section provides information on the following subjects:

- [Understanding WebLogic JAM Connectivity](#)
- [Step 1: Deploy WebLogic JAM in the WebLogic Server Environment](#)
- [Step 2: Define Where the CRM Will Run](#)
- [Step 3: Create a Logical Unit for the CRM](#)
- [Step 4: Connect the CRM to Back-End Systems on the Mainframe](#)
- [Step 5: Enter Connectivity Information into WebLogic Administration Console](#)
- [Step 6: Define a WebLogic JAM Gateway](#)
- [Step 7: Verify Your WebLogic JAM Connectivity Configuration](#)

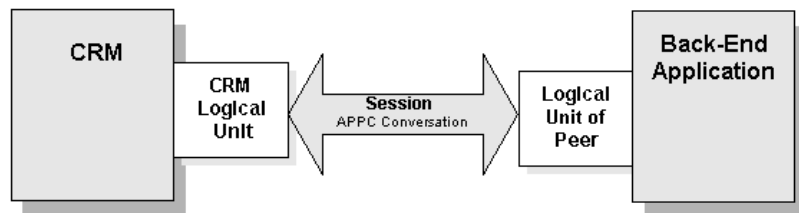
# Understanding WebLogic JAM Connectivity

WebLogic JAM uses two distributed software components to connect to your back-end systems: the WebLogic JAM Gateway and the Communications Resource Manager (CRM). The WebLogic JAM Gateway component runs within an instance of WebLogic Server and serves as a proxy to other applications running within WebLogic Server.

The CRM runs as a native operating system process, and it connects to your back-end system using Advanced Program to Program Communications (APPC), also known as LU 6.2, using SNA network connections. The CRM and the WebLogic JAM Gateway communicate with each other using a TCP/IP socket. The CRM connects to your back-end system using an SNA network connection called a logical unit (LU).

A logical unit is an SNA network's way of providing access to the SNA network to end users and software programs. A logical unit is a unique, addressable part of an SNA network that manages data flows between network partners. A logical unit is somewhat like a TCP/IP address and port because software programs can use it to access the network and to communicate with other software programs that are distributed throughout the enterprise. Unlike TCP/IP connections, logical units must be defined prior to use. Figure 2-1 shows the CRM using APPC to establish communication with a back-end application.

**Figure 2-1 CRM Using APPC**



In order for software programs such as the CRM to communicate via APPC, each peer program must have access to a logical unit. The software programs then allocate a session between the two logical units to communicate and collaborate.

A session is a pipeline between two logical units that manages the exchange of data between the logical units. The two peer programs send data back and forth over the pair of logical units using a session. This exchange of data is called a conversation. The number of sessions that can exist simultaneously over a given logical unit pair is configured in the SNA network. Establishing WebLogic JAM connectivity involves allocating logical units and sessions in your SNA network, and recording this configuration in WebLogic JAM.

The WebLogic Administration Console allows you to define where WebLogic JAM components will run within your enterprise, and the network connections that they will establish. Once entered into the console, this configuration is persisted and distributed to WebLogic JAM components upon start-up.

## Getting Started with WebLogic JAM Connectivity

The overall task of establishing WebLogic JAM connectivity involves asking administration personnel to allocate SNA network resources (logical units, sessions) and then recording these resources in WebLogic JAM configuration via the WebLogic Administration Console. This task has been organized into six primary steps:

- [Step 1: Deploy WebLogic JAM in the WebLogic Server Environment](#)
- [Step 2: Define Where the CRM Will Run](#)
- [Step 3: Create a Logical Unit for the CRM](#)
- [Step 4: Connect the CRM to Back-End Systems on the Mainframe](#)
- [Step 5: Enter Connectivity Information into WebLogic Administration Console](#)
- [Step 6: Define a WebLogic JAM Gateway](#)
- [Step 7: Verify Your WebLogic JAM Connectivity Configuration](#)

Your system administrators will configure your mainframe to communicate using SNA and then establish the actual connection to the CRM using the parameter information provided in the steps of this section.

When you have all of the appropriate parameters, you are ready to configure connectivity by entering them into the WebLogic Administration Console. Use the steps in this section to help you and your system administrators connect your systems

correctly and to get the correct configuration information into the WebLogic Administration Console. Instructions are also provided to help you verify your configuration.

**Note:** The Mainframe Connectivity Worksheet is provided to help you prepare to configure connectivity between your mainframe system and your BEA WebLogic Server system. For a copy of the worksheet, see [Appendix A, “Mainframe Connectivity Worksheet.”](#)

# Step 1: Deploy WebLogic JAM in the WebLogic Server Environment

After WebLogic JAM has been installed, it must be deployed in the WebLogic Server environment. The following tasks must be completed if you are not running a pre-configured sample.

You can deploy WebLogic JAM in the following ways:

- If you are creating applications, the `example` domain and the `verify` domain are examples of domains deployed in the WebLogic Server environment. You may want to use these domains as a starting point, adding other J2EE applications to meet your specific requirements.
- If you are adding WebLogic JAM to existing WebLogic Server applications, you need to manually deploy WebLogic JAM by completing the following steps:
  - [Adding the jam.jar File to the WebLogic Server CLASSPATH](#)
  - [Deploying the jam.ear File](#)
  - [Defining the WebLogic JAM Startup Class](#)



## Adding the jam.jar File to the WebLogic Server CLASSPATH

The `jam.jar` file adds classes to the WebLogic Server **CLASSPATH**. The `jam.jar` file must be added to the **CLASSPATH** in the WebLogic Server startup script for each server where it will run.

To manually add the `jam.jar` file to the WebLogic Server **CLASSPATH**, perform the following steps:

1. Using the text editor of your choice, locate the script you use to start WebLogic Server in your domain. For example, use the `startWebLogic` script in your `config/mydomain` directory where `mydomain` is the name of your domain.
  - For example, on a Microsoft Windows platform, use your `startWebLogic.cmd` script in your `config/mydomain` directory where `mydomain` is the name of your domain and `startWebLogic.cmd` is the script to start WebLogic Server.
  - For example, on a Unix platform, use your `startWebLogic.sh` script in your `config/mydomain` directory where `mydomain` is the name of your domain and `startWebLogic.sh` is the script to start WebLogic Server.
2. Edit the `startWebLogic` script by adding the `jam.jar` file to the **CLASSPATH** setting as shown in the following examples:
  - For Microsoft Windows:

```
set CLASSPATH=.;.\lib\weblogic_sp.jar;
.\lib\weblogic.jar;e:\bea\wljam5.1\lib\jam.jar
```

In this example, `e:\bea\wljam5.1` is the install directory for WebLogic JAM.
  - For Unix:

```
CLASSPATH=$WL_HOME:$WL_HOME/lib/weblogic_sp.jar:$WL_HOME/lib
/weblogic.jar:./ext/weblogic-tags.jar:/bea/wljam5.1/lib/jam.
jar
```

In this example, `/bea/wljam5.1` is the install directory for WebLogic JAM.
3. Save the file and close the text editor.

# Deploying the jam.ear File

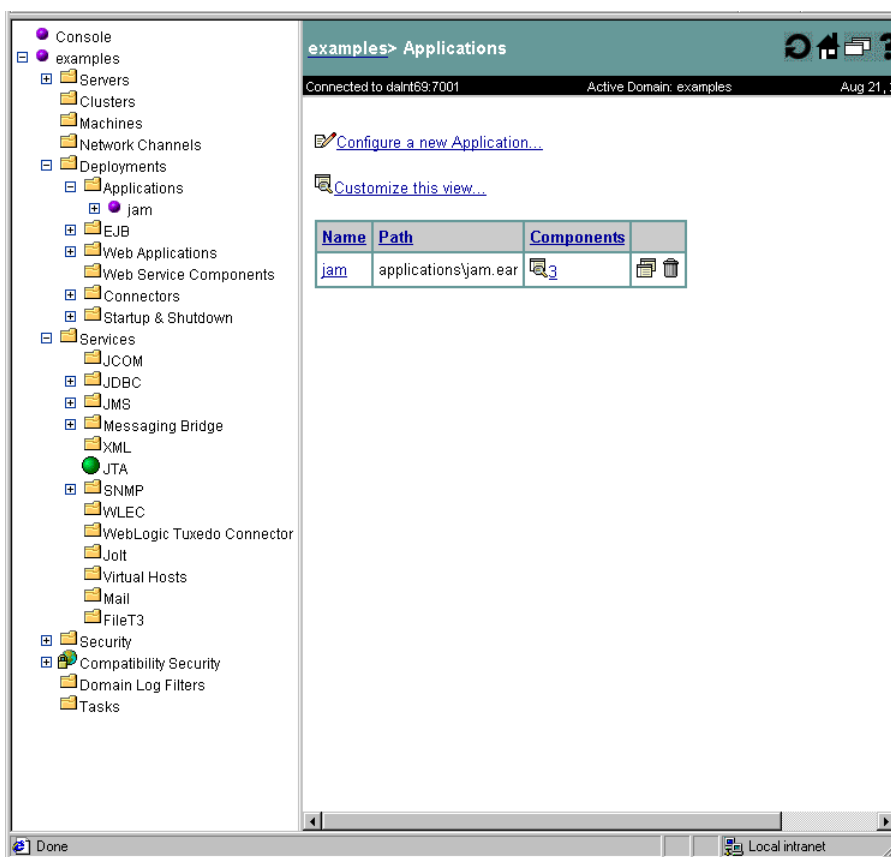
The jam.ear file deploys the WebLogic JAM applications and the WebLogic JAM extension to the WebLogic Administration Console. This file contains the following EJBs:

- jambeans
- jamtojms
- jam.war (for the console)

## Step 1: Deploy WebLogic JAM in the WebLogic Server Environment

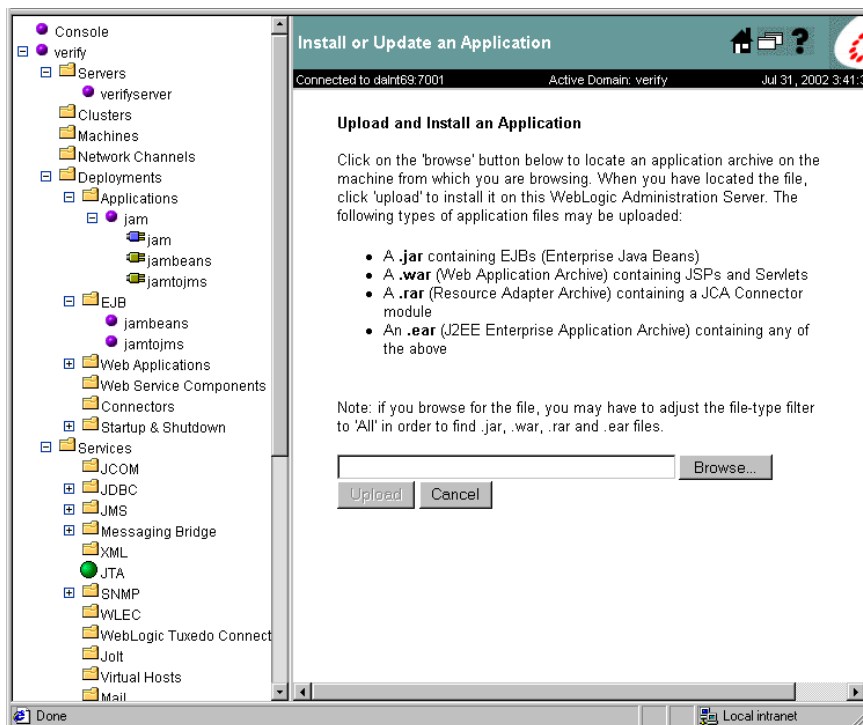
Deploy the `jam.ear` file on the appropriate servers using the WebLogic Administration Console. To manually deploy the `jam.ear` file, perform the following steps:

1. In the WebLogic Administration Console, select **Deployments**→**Applications** to invoke the **Applications** page. Click on **Configure a new Application**.



## 2 Configuring WebLogic JAM Connectivity

2. Click **upload it through your browser**. Click **Browse** to locate the `jam.ear` file in `wljam5.1\lib` directory. Click **Upload**.



**Note:** The Path field and the Browse button specify directories and files relative to the computer that the browser is running on.

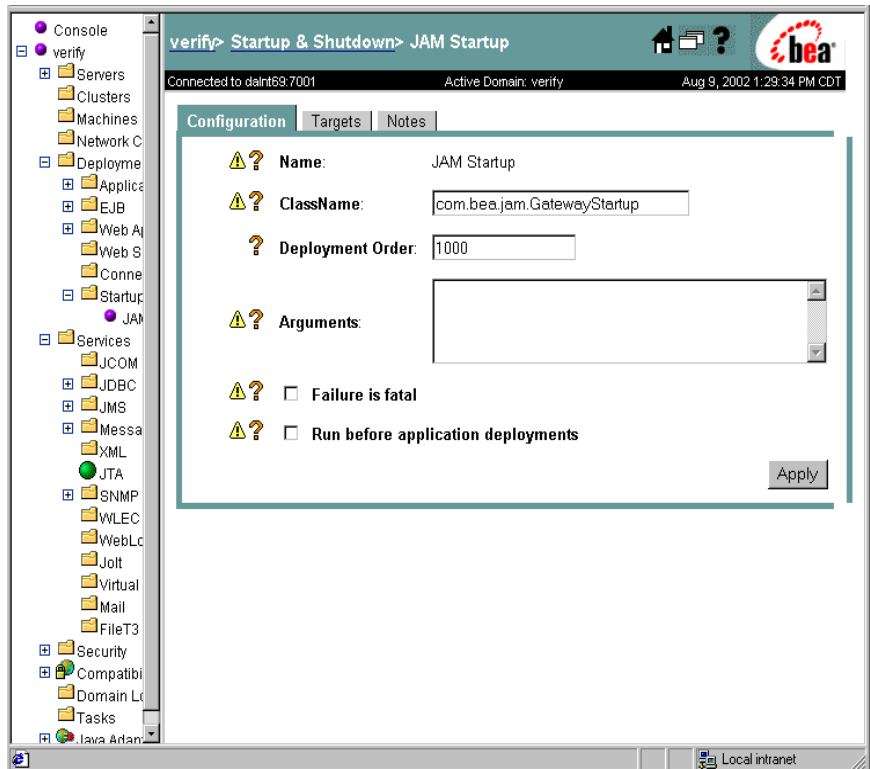
The `jam.ear` file has been uploaded and WebLogic JAM applications and components are available in the left pane of the WebLogic Administration Console.

## Defining the WebLogic JAM Startup Class

The WebLogic JAM startup class defines a class that is launched at WebLogic Server startup. To manually define the WebLogic JAM startup class using the WebLogic Administration Console, perform the following steps.

## Step 1: Deploy WebLogic JAM in the WebLogic Server Environment

1. In the WebLogic Administration Console, select **Deployments**→**Startup & Shutdown** to invoke the **Startup & Shutdown** page.
2. Click on **Configure a new Startup Class** to invoke the **Create a new Startup Class** page.
3. Fill in the fields.
  - a. Assign a name for the **Name** field.
  - b. The **ClassName** must be `com.bea.jam.GatewayStartup`.
4. Click **Create** to create the new startup class.



5. Select the **Targets** tab. Select the target server from the available list and click the arrow to add it to the chosen list. Click **Apply** to apply the new startup class.

## Step 2: Define Where the CRM Will Run

The CRM should run on a mainframe that can access the desired back-end systems using your SNA network. The CRM can run on the mainframe as a job in an MVS region or as a process under z/OS or OS/390 Unix using Unix System Services (USS). As part of the WebLogic JAM installation process, you are instructed on how to install the CRM. For detailed information, see the *BEA WebLogic JAM Installation Guide*.

For this step, you need to determine where you want the CRM to run and then gather the configuration information that you will input into the WebLogic Administration Console in a later step. [Table 2-1](#) provides a list of the parameters that you will use to configure connectivity with the CRM in the WebLogic Administration Console.

**Table 2-1 CRM Definition Parameter in the WebLogic Administration Console**

<b>WebLogic Administration Console Parameters</b>	<b>Description</b>	<b>Parameter Syntax</b>
Name	The arbitrary name of the CRM. The CRM process uses this name to ensure the correct configuration between the WebLogic JAM Gateway and the CRM process.	Alpha-numeric string Ex. CRM1
Listen Address	The host name of the machine where the CRM runs, or the TCP/IP address of the machine where the CRM runs.	Alpha-numeric string or dotted IP address Ex. myhost or 123.4.5.678
Listen Port	The TCP/IP port that is used by the CRM to listen for connecting WebLogic JAM components. Ask your network administrator for an unused TCP/IP port on the machine where the CRM runs.	Numeric value Ex. 8002

WebLogic Administration Console Parameters	Description	Parameter Syntax
Stack Type	The predefined name that the CRM uses to identify the supported SNA stack software release. Refer to the <i>BEA WebLogic Java Adapter for Mainframe Release Notes</i> for supported stacks.	Supported stack available in drop-down list. Ex. vtm28

**Note:** A worksheet is available to give you a central location for gathering various connection parameters before you sit down to enter them into the WebLogic Administration Console in [“Step 5: Enter Connectivity Information into WebLogic Administration Console.”](#) For a copy of the worksheet, see [“Mainframe Connectivity Worksheet.”](#)

## Step 3: Create a Logical Unit for the CRM

The CRM component of WebLogic JAM uses an LUTYPE 6.2 logical unit to communicate with back-end systems running on the mainframe. Each back-end system must have its own logical unit, and sessions must be defined to allow the CRM to allocate sessions between its logical unit and the logical unit of the mainframe system. Each peer must be configured ahead of time to connect to the logical unit of its peer.

In this step, your VTAM administrator creates a logical unit within VTAM for the CRM. The logical unit defines the CRM to the SNA network. You will need to enter this logical unit parameter into the WebLogic Administration Console to finish WebLogic JAM configuration in a later step.

This step provides the following information:

- [Parameters for Establishing the CRM Logical Unit](#)

Provides the parameters that the VTAM administrator needs to create a logical unit for the CRM

- [WebLogic Administration Console Parameter for the CRM Logical Unit](#)

Provides the parameter that you need to enter into the WebLogic Administration Console to complete WebLogic JAM configuration as described in a later step.

# Parameters for Establishing the CRM Logical Unit

The VTAM application program major node (VTAM APPLID) is also the CRM logical unit. You define the APPLID to VTAM and activate it in the VTAM network.

[Listing 2-1](#) shows a sample VTAM APPLID definition node.

### Listing 2-1 Sample VTAM APPLID Definition Node:

---

```
BEA    VBUILD TYPE=APPL                APPLICATION MAJOR NODE
BEAAPPL1 APPL  ACBNAME=BEAAPPL1,        ACBNAME FOR APPC
                APPC=YES,
                SYNCLVL=SYNCPT,
                PARSESS=YES
```

---

The following parameters represent the minimum requirements for defining the logical unit.

#### **Application Program Major Node Name**

This tag identifies the node to the network.

#### **TYPE=APPL**

Major node should be an Application Major Node type.

#### **ACBNAME=LU Name**

Logical unit name. This can be identical to the Application Major Node name. This is the logical unit name used in the CRM configuration.

#### **APPC=YES**

Defines this as an APPC logical unit.

#### **SYNCLVL=[CONFIRM | SYNCPT]**

Determines level of transactional support. CONFIRM supports non-transactional DPL and APPC service requests only, while SYNCPT will in addition, support transactional service requests. See [“Administering Transactions”](#) for information on these options.



**PARSESS=YES**

Allows multiple sessions between session partners.

**SRBEXIT=NO**

The CRM logical unit does not support the service request block (SRB) exit routines. This parameter must be set to NO (the default value if the parameter is not specified.)

## WebLogic Administration Console Parameter for the CRM Logical Unit

[Table 2-2](#) shows the parameter that you will use in the WebLogic Administration Console to configure connectivity with the CRM. Work closely with your VTAM administrator to get the CRM Logical Unit defined in your SNA network.

**Table 2-2 CRM Logical Unit Definition Parameter in the WebLogic Administration Console**

WebLogic Administration Console Parameter	Description	Parameter Syntax
Logical Unit	The name of the logical unit defined in VTAM for the CRM.	Alpha-numeric string Ex. BEAAPPL1

**Note:** A worksheet is available to give you a central location for gathering various connection parameters before you sit down to enter them into the WebLogic Administration Console in [“Step 5: Enter Connectivity Information into WebLogic Administration Console.”](#) For a copy of the worksheet, see [“Mainframe Connectivity Worksheet.”](#)

# Step 4: Connect the CRM to Back-End Systems on the Mainframe

To connect the CRM to your mainframe applications, you may need to connect to different types of regions that run on the mainframe. Instances of IMS and CICS systems are usually referred to as an IMS region or a CICS region respectively. APPC applications that do not run in CICS or IMS, but rather run under APPC/MVS as a started job, can also connect to WebLogic JAM. These applications are referred to in WebLogic JAM documentation as running in a batch region.

This section provides information on the following subjects:

- [Determining Connection Characteristics](#)
- [Connecting to a CICS Region](#)
- [Connecting to an IMS Region](#)
- [Connecting to a Batch Region](#)

## Determining Connection Characteristics

When you define the connectivity of the CRM to the back-end application environment, you must identify characteristics of this connection. Connection characteristics determine the type of work and class of service used between the CRM and its partner. Speak with your system administrator ahead of time about these considerations so you may select the best choices for your configuration.

In order to connect to a back-end system using WebLogic JAM, you must configure your CRM with an APPC connection (or link) to the back-end system. These connections are referred to in WebLogic JAM as CRM Links. A CRM Link is a definition that WebLogic JAM uses to store the parameters of the connection (or link) between the CRM's logical unit and the logical unit of the back-end system. A CRM can only have a single CRM Link to a given region.

Discuss the following list of topics with your system administrator. These topics are relevant to both the CRM and back-end application environment.

- [Logmode Name](#)
- [Sessions](#)
- [Security Level](#)

### Logmode Name

Mode name defines characteristics of the APPC sessions that the CRM establishes across this CRM Link.

The characteristics of different logon modes are tailored to support different types of applications. For example long-running batch applications might use different logon modes than short-lived online applications. Logon modes can define different logical unit protocols, classes of service, packet sizes, and pacing algorithms.

Use the same logon mode for both the CRM logical unit and the back-end application logical unit.

### Sessions

When a CRM Link is created, a pool of sessions is allocated to handle requests for that link. The sessions in the pool are available for requests that originate on either side. The CRM requires a session for each request originating in WebLogic Server, while the back-end system requires a session for each request originating there. If a session is not available to a request, then the request will fail.

The following topics should be considered for setting up session connections:

- **Maximum Session**

The term “maximum sessions” refers to the number of sessions in the allocated pool. This value determines the number of concurrent requests that can be active at any given time. Both the CRM and the back-end system must configure their maximum sessions to the same value.

- **Minimum Winner Sessions**

Both the CRM and the back-end system are pre-allocated a number of sessions for their use. This number is referred to as “minimum winner sessions.” The owner of these pre-allocated sessions has priority for its winner sessions; however, they may be reassigned depending on system load.

The total of the minimum sessions for both sides must be less than or equal to the maximum session value. If the total is higher than the maximum session value, you will be unable to activate the link.

For best results in determining minimum sessions, evaluate the number of sessions that are required for the CRM and for the back-end system to support concurrent requests.

### Security Level

The security level of a CRM Link defines what security credentials are required for all requests that utilize this link.

- No security credentials provided

In the simplest case, no security credentials are provided with requests sent on the link. This implies that the user has been authenticated in the originating domain, and that no further authentication is required. The value for the CRM would be `Local`. The corresponding value for the back-end system must be defined in the connection definitions.

- User ID provided

It is possible that the application processing the request needs to identify the user making the request for access control to application resources. For this case, only a user ID is provided with the request. The value for the CRM would be `Identify`. The corresponding value for the back-end system must be defined in the connection definitions.

- User ID and password provided

A third possibility is that full security credentials are required for any requests sent on the link. In this case, a user ID and password are required for each request. These credentials are validated against the security configuration in the destination environment. The user ID can then be used for access control to application resources. The value for the CRM would be `Verify`. The corresponding value for the back-end system must be defined in the connection definitions.

The following sections describe the steps necessary to connect to each type of back-end system. Configuration that is required on the mainframe system is discussed, as well as capturing the parameters of the CRM Link that will be used to connect to the mainframe system. Each specific type of back-end system is discussed separately below.

## Connecting to a CICS Region

Connecting to a CICS region is a process that requires you to work closely with your system administrator to establish the actual connection. First, you and your administrator define the CRM's logical unit to the CICS region with a CICS Connection Definition. Then you must define the connection characteristics of the CRM's connection to the CICS region with a CICS Session Definition. Once the connection has been defined in the CICS region, you need to enter the information into the WebLogic Administration Console as a CICS region definition and a CRM Link definition. This section provides information about the following:

- [Parameters and Steps for Connecting to a CICS Region](#)

A system administrator will most likely configure the CICS region and establish the CICS connections to the CRM. This section provides information to help the administrator establish those connections.

- [WebLogic Administration Console Parameters for CICS Region Connectivity](#)

You will need certain parameters created during the connectivity process when you sit down at the WebLogic Administration Console to finish WebLogic JAM configuration. This section provides information to help you understand and capture the parameters you will need to enter into the WebLogic Administration Console in a later step.

## Parameters and Steps for Connecting to a CICS Region

For every CICS region that the CRM connects to, resource definitions for the CRM's logical unit must be configured in the CICS region. This configuration consists of CICS connection and session resource definitions. These CICS resource definitions must be installed in each CICS region to which the CRM connects. Use the following steps and parameters to establish CICS region connectivity:

- [Step 1: Create a Logical Unit for the CICS Region](#)
- [Step 2: Add APPC to the CICS Region](#)
- [Step 3: Define the CICS Connection Definition](#)
- [Step 4: Define the CICS Session Definition](#)
- [Step 5: Install the CICS Connection and Session Definitions](#)

## 2 *Configuring WebLogic JAM Connectivity*

---

### ■ [Step 6: Verify Connection and Session Status](#)

#### Step 1: Create a Logical Unit for the CICS Region

The CICS application program major node is also the CICS logical unit. This application program major node is defined to VTAM when the CICS region is created. Ask your system administrator to provide the CICS VTAM APPLID of the created CICS region.

#### Step 2: Add APPC to the CICS Region

To enable the CICS region for APPC, start the region with the following system initialization parameter:

`ISC=YES`

This enables inter-system communication.

#### Step 3: Define the CICS Connection Definition

In each CICS region to which the CRM connects, a system administrator must define and install a connection resource definition and session resource definition in the region. The connection and session definitions should be defined in the same resource group. This group of definitions defines the CRM as a partner Logical Unit to the CICS region, and associates connection characteristics to the link between the CRM and CICS. Listing 2-2 shows an example CICS connection definition.

The following parameters represent the minimum requirements for defining the CICS connection.

**ACCESSMETHOD (VTAM)**

Indicates this resource is accessed via VTAM

**PROTOCOL (APPC)**

Indicates that this connection is used for APPC

**NETNAME (xxx)**

The name of the CRM's logical unit created in VTAM.

**ATTACHSEC (xxx)**

The security protocol used for this connection. Set to `LOCAL`, `IDENTIFY`, or `VERIFY`. If security credentials are not passed between WebLogic Server and CICS, set this to `LOCAL`. If user IDs are required between WebLogic Server

## Step 4: Connect the CRM to Back-End Systems on the Mainframe

---

and CICS set this parameter to `IDENTIFY`. If user IDs and passwords are required, set this parameter to `VERIFY`.

### **AUTOCONNECT (NO)**

Sessions for this connection should not be automatically initialized when CICS is started. The CRM will initiate session acquisition.

### **Listing 2-2 Example CICS Connection Definition**

---

```
DEFINE CONNECTION(BEA1)
  GROUP(BEA)
  DESCRIPTION(LOGICAL UNIT FOR CRM1 TO ACCESS THIS CICS)
  ACCESSMETHOD(VTAM)
  PROTOCOL(APPC)
  NETNAME(BEAAPPL1)
  ATTACHSEC(LOCAL)
  AUTOCONNECT(NO)
```

---

## Step 4: Define the CICS Session Definition

In each CICS region that the CRM connects to, a session resource definition must also be defined and installed in the region. Listing 2-3 shows an example CICS connection definition. The following parameters represent the minimum requirements for defining the CICS session:

### **CONNECTION (xxx)**

Should be set to the CICS connection definition for CRM's logical unit.

### **AUTOCONNECT (YES)**

Determines how the activation of the session is negotiated. `YES` enables the CICS/ESA host to negotiate its own winner sessions once the CRM starts the initiation of session acquisition.

### **MODENAME (xxx)**

This is the VTAM LOGMODE name used for these sessions.

### **MAXIMUM(SESSIONBR, WINNER)**

Defines the total number of sessions for this connection, and the number that are established by the CICS region. The `SESSIONBR` number defines the total number of sessions allowed for this connection. The `WINNER` number defines the sessions that are established by the CICS region. It is recommended that

## 2 *Configuring WebLogic JAM Connectivity*

---

you set the WINNER numbers to be the remainder of the MAXIMUM SESSIONS after you define the number of WINNER numbers needed by the CRM.

### **Listing 2-3 Example CICS Session Definition**

---

```
DEFINE SESSION(BEAL)
  GROUP(BEA)
  CONNECTION(BEAL)
  DESCRIPTION(SESSIONS FOR CRM1 TO ACCESS THIS CICS)
  PROTOCOL(APPC)
  AUTOCONNECT(YES)
  MODENAME(SMSNA100)
  MAXIMUM(20,10)
```

---

### Step 5: Install the CICS Connection and Session Definitions

To install the resource definitions, put them on the host in a separate group. You may use the CEDA INSTALL command to install the group to CICS.

For example:

```
CEDA INSTALL GROUP(BEA)
```

**Note:** The CRM cannot connect to the CICS region until the resource definitions are installed and ready for service by CICS.

### Step 6: Verify Connection and Session Status

After you have installed the resource definitions, you can verify the status of connections and sessions using the CICS/ESA system commands shown in Listing 2-4.

### **Listing 2-4 Example Commands for Viewing Connection and Session Status**

---

```
CEMT I CONN(BEAL) **View the status of the connection
CEMT I NET(BEAAPPL1) **View the status of the sessions
CEMT I MODENAME(SMSNA100) **View the status of the mode
```

---



## WebLogic Administration Console Parameters for CICS Region Connectivity

When you configure WebLogic JAM connectivity for a CICS region in the WebLogic Administration Console, you will need to supply the following parameters:

- [CICS Region](#)
- [CRM Link for a CICS Region](#)

**Note:** A worksheet is available to give you a central location for gathering various connection parameters before you sit down to enter them into the WebLogic Administration Console in “[Step 5: Enter Connectivity Information into WebLogic Administration Console.](#)” For a copy of the worksheet, see “[Mainframe Connectivity Worksheet.](#)”

### CICS Region

[Table 2-3](#) shows the parameters that you will use in the WebLogic Administration Console to configure a CICS region.

**Table 2-3 CICS Region Logical Unit Definition Parameters in the WebLogic Administration Console**

WebLogic Administration Console Parameters	Description	Parameter Syntax
Name	The arbitrary name of the region to which the CRM connects.	Alpha-numeric string Ex. CICS3
Logical Unit	The CICS logical unit name. The application identifier (VTAMAPPLID) of the CICS region to which the CRM connects.	Eight character alpha-numeric string Ex. CICSAPPL

### CRM Link for a CICS Region

Table 2-4 shows the parameters that you will use in the WebLogic Administration Console to configure a CRM Link for a CICS region. WebLogic JAM uses a CRM Link definition to store the parameters of the connection (or link) between the CRM's logical unit and the logical unit of the back-end system. These parameters determine the characteristics of the logical connection made between the CRM and the CICS region.

These values override values provided in the CRM VTAM application definition. For information on determining these values, see “[Determining Connection Characteristics](#).” For specific values, ask your system administrator to retrieve the values for your CICS connection and session definitions.

**Table 2-4 CICS Region CRM Link Definition Parameters in the WebLogic Administration Console**

WebLogic Administration Console Parameters	Description	Parameter Syntax
Name	The arbitrary name of the CRM Link for this CICS region.	A nine character alpha-numeric string Ex. CRM1Link
Region Name	The name of the CICS region for which you are supplying this link. This should correspond to the arbitrary name provided for the CICS region.	Alpha-numeric string Ex. CICS3
Logmode Name	The class of services for this link. This should be the MODENAME on the CICS session definition.	Alpha-numeric string Ex. SMSNA100
Maximum Sessions	Defines the total number of sessions for this link. The first value of MAXIMUM (SESSNBR, WINNER) from the CICS session definition.	Numeric value Ex. 20

WebLogic Administration Console Parameters	Description	Parameter Syntax
Minimum Winner Sessions	The number of winner sessions pre-allocated to the CRM for this link.  This value should be less than or equal to the maximum session count minus the second value of MAXIMUM(SESSNBR, WINNER) from the CICS session definition.	Numeric value Ex. 10
Security Level	The security credentials required for requests on this link.  This should match the parameter value provided in the ATTACHSEC() parameter of the CICS connection definition.	Supported values available in list (Local, Identify, Verify). Ex. Local
Default User ID	A user ID to be provided for requests from applications that do not provide security credentials.  This parameter is for use only with Security Level IDENTIFY.	Alpha-numeric string Ex. USER01

## Connecting to an IMS Region

Connecting to an IMS region is a process that requires you to work closely with your system administrator to establish the actual connection. In order for the CRM to connect to IMS, APPC/MVS must be installed and started, and APPC/MVS must be configured with a logical unit that points to the IMS region. Then you need to gather information about that connection to input into the WebLogic Administration Console in a later step. This section provides information about the following:

- [Parameters and Steps for Connecting to an IMS Region](#)

A system administrator will most likely configure the IMS region. This section details how to configure VTAM, APPC/MVS and an IMS region to allow the CRM to connect to the IMS region via APPC.

- [WebLogic Administration Console Parameters for IMS Region Connectivity](#)

You will need certain parameters created during the connectivity process to complete the WebLogic JAM configuration using the WebLogic Administration Console. This section provides information to help you understand and capture the parameters you will need to enter into the WebLogic Administration Console.

### Parameters and Steps for Connecting to an IMS Region

Use the following steps and parameters to establish connectivity for an IMS region:

- [Step 1: Create a Logical Unit for an IMS Region and Add it to APPC/MVS](#)
- [Step 2: Add APPC to Your IMS Region](#)

#### Step 1: Create a Logical Unit for an IMS Region and Add it to APPC/MVS

You must create a logical unit for the IMS region. In addition to each IMS region having its own logical unit, sessions must be defined that allow the CRM to allocate sessions between its logical unit and the logical unit of the IMS region.

[Listing 2-5](#) provides a sample VTAM definition of a logical unit for an IMS region and its application major node:

#### **Listing 2-5 VTAM Definition of a Logical Unit for an IMS Region**

---

```
IMS VBUILD TYPE=APPL                                APPLICATION MAJOR NODE
IMSAPPL1 APPL    ACBNAME=IMSAPPL1 ,                ACBNAME FOR APPC
                APPC=YES ,
                DLOGMOD=SMSNA100 ,
                PARSESS=YES ,
                SRBEXIT=YES ,
                DSESLIM=10 ,
                DMINWNL=5 ,
                SECACPT=NONE ,
                SYNCLVL=SYNCPT
```

---

The following parameters represent the minimum requirements for creating a VTAM APPLID for an IMS region.

## Step 4: Connect the CRM to Back-End Systems on the Mainframe

---

### **Application Program Major Node Name**

This tag identifies the node to the network.

### **TYPE=APPL**

A major node should be an Application Major Node.

### **ACBNAME=LU Name**

Logical unit name. This can be identical to the Application Major Node name. This is the name used for the IMS logical unit.

### **APPC=YES**

Defines an APPC logical unit.

### **DLOGMOD=xxx**

This is the VTAM LOGMODE name used for these sessions. An example is SMSNA100, which is the default recommended for WebLogic JAM.

### **PARSESS=YES**

Allows multiple sessions between session partners.

### **SECACPT=[NONE | ALREADYV | CONV]**

This parameter sets the level of security and must match a corresponding value for the CRM.

**NONE** means no security credentials are provided with requests sent on the link.

**ALREADYV** means that the application processing the request needs to identify the user making the request for access control to application resources. A user ID is provided with the request.

**CONV** means that full security credentials are required for any requests sent on the link. A user ID and password are required for each request.

### **SYNCLVL=[CONFIRM | SYNCPT]**

Determines level of transactional support. **CONFIRM** supports non-transactional DPL and APPC service requests only, while **SYNCPT** will in addition, support transactional service requests. See [“Administering Transactions”](#) for information on these options.

### **DSESLIM=n**

Defines the total number of sessions for this connection, and the number that are established by the IMS region.

### **DMINWNL=n**

Defines the sessions that are established by the IMS region. It is recommended that you set the **WINNER** numbers to be the remainder of the

## 2 Configuring WebLogic JAM Connectivity

---

MAXIMUM SESSIONS after you define the number of WINNER numbers needed by the CRM.

After you create an IMS logical unit (VTAM APPLID), you must add it to APPC/MVS. [Listing 2-6](#) is a sample for adding the IMS logical unit to APPC/MVS.

### Listing 2-6 Sample Command for Associating the Logical Unit with an IMS Region

---

```
LUADD ACBNAME(IMSAPPL1) BASE SCHED(IMSREG) TPDATA(SYS1.APPCTP)
SIDEINFO DATASET(SYS1.APPCSI)
```

---

The following parameters represent the minimum requirements for adding a logical unit to APPC/MVS.

**ACBNAME( xxx )**

This should match the ACBNAME of the IMS logical unit's VTAM APPLID.

**SCHED( xxx )**

This specifies that the IMS region is the scheduler for all requests on this logical unit.

### Step 2: Add APPC to Your IMS Region

Once APPC/MVS has been set up to schedule requests to your IMS region, IMS must also be configured to activate APPC within IMS. To activate APPC in IMS you must specify APPC=Y in the IMS start-up procedure or in IMS.PROCLIB member DFSPBxxx where xxx is your RGSUF= parameter.

[Listing 2-7](#) shows an example of the DFSPBxxx member. [Listing 2-8](#) shows an example of the IMS PROC member.

### Listing 2-7 Example DFSPBxxx member (entire member not shown)

---

```
APPC=Y,
RES=Y,
FRE=00030,
QBUF=0005,
```

---

### Listing 2-8 Example IMS PROC (entire member not shown)

---

```
//          PROC RGN=2000K,SOUT=A,DPTY='(14,15)',
//          SYS=,SYS1=,SYS2=,
//          RGSUF=IV1,PARM1=APPC=Y,PARM2=,APPLID1=IMS61CR1,AOIS=R
//IEFPROC EXEC PGM=DFSVMRC0,DPRTY=&DPTY,
//          REGION=&RGN,
//          PARM='CTL,&RGSUF,&PARM1,&PARM2,&APPLID1,&AOIS'
```

---

## WebLogic Administration Console Parameters for IMS Region Connectivity

When you configure WebLogic JAM connectivity for an IMS region in the WebLogic Administration Console, you will need to supply the following parameters:

- [IMS Region Logical Unit](#)
- [IMS Region CRM Link](#)

**Note:** A worksheet is available to give you a central location for gathering various connection parameters before you sit down to enter them into the WebLogic Administration Console in [“Step 5: Enter Connectivity Information into WebLogic Administration Console.”](#) For a copy of the worksheet, see [“Mainframe Connectivity Worksheet.”](#)

### IMS Region Logical Unit

[Table 2-5](#) shows the parameters to use in the WebLogic Administration Console to configure IMS region’s logical unit. Work closely with your administrator to get the CRM logical unit defined in your SNA network.

## 2 Configuring WebLogic JAM Connectivity

**Table 2-5 IMS Region Logical Unit Definition Parameters in the WebLogic Administration Console**

<b>WebLogic Administration Console Parameters</b>	<b>Description</b>	<b>Parameter Syntax</b>
Name	The arbitrary name of the region to which the CRM connects.	Alpha-numeric string Ex. IMS1
Logical Unit	The IMS logical unit name. The logical unit ID of the APPC/MVS LU for the IMS region to which the CRM connects.	Eight character alpha-numeric string Ex. IMSAPPL1

### IMS Region CRM Link

[Table 2-6](#) shows the parameters to use in the WebLogic Administration Console to configure IMS region's CRM Link. WebLogic JAM uses a CRM Link definition to store the parameters of the connection (or link) between the CRM's logical unit and the logical unit of the back-end system. These parameters determine the characteristics of the logical connection made between the CRM and the IMS region.

**Table 2-6 IMS Region CRM Link Definition Parameters in the WebLogic Administration Console**

<b>WebLogic Administration Console Parameters</b>	<b>Description</b>	<b>Parameter Syntax</b>
Name	The arbitrary name of the CRM Link for this IMS region.	Alpha-numeric string Ex. CRM1IMS1
Region Name	The name of the IMS region for this link. This should correspond to the arbitrary name provided for the IMS region.	Alpha-numeric string Ex. IMS1



## Step 4: Connect the CRM to Back-End Systems on the Mainframe

WebLogic Administration Console Parameters	Description	Parameter Syntax
Logmode Name	The class of services for this link. This should be the DLOGMOD on the IMS session definition.	Alpha-numeric string Ex. SMSNA100
Maximum Sessions	Defines the total number of sessions for this link. The value of DSESLIM from the IMS logical unit definition.	Numeric value Ex. 20
Minimum Winner Sessions	The number of winner sessions pre-allocated to the CRM for this link. This value should be less than or equal to the maximum session count minus the value of DMINWNL from the IMS logical unit definition.	Numeric value Ex. 10
Security Level	The security credentials required for requests on this link. This should be compatible with the corresponding value provided in the SECACPT(NONE   ALREADYV   CONV) parameter of the IMS connection definition. For more information, see <a href="#">“Step 1: Create a Logical Unit for an IMS Region and Add it to APPC/MVS.”</a>	Supported values available in drop-down list (Local, Identify, Verify). Ex. Local
Default User ID	A user ID to be provided for requests from applications that do not provide security credentials. This parameter is for use only with Security Level IDENTIFY.	Alpha-numeric string Ex. USER01

# Connecting to a Batch Region

Connecting to a batch region is a process that requires you to work closely with your system administrator to establish the actual connection. WebLogic JAM can connect to APPC applications that do not run in CICS or IMS, but rather run in MVS as a started job and use APPC/MVS.

## Parameters and Steps for Connecting to a Batch Region

Use the following steps and parameters to establish connectivity for a batch region:

- [Step 1: Create a Logical Unit for a Batch Region and Add it to APPC/MVS](#)
- [Step 2: Add a Logical Unit for a Batch Region to APPC/MVS](#)

### Step 1: Create a Logical Unit for a Batch Region and Add it to APPC/MVS

You must create a logical unit for the batch region. In addition to each batch region having its own logical unit, sessions must be defined that allow the CRM to allocate sessions between its logical unit and the logical unit of the batch region.

[Listing 2-9](#) provides a sample VTAM definition of a logical unit for a batch region and its application major node:

#### Listing 2-9 VTAM Definition of a Logical Unit for a Batch Region

---

```
BATCH1 VBUILD TYPE=APPL                                APPLICATION MAJOR NODE
MVSAPPL1 APPL  ACBNAME=MVSAPPL1 ,                      ACBNAME FOR APPC
              APPC=YES ,
              DLOGMOD=SMSNA100 ,
              PARSESS=YES ,
              SRBEXIT=YES
              DSESLIM=10 ,
              DMINWNL=5 ,
              SECACPT=NONE ,
              SYNCLVL=SYNCPT
```

---

The following parameters represent the minimum requirements for creating a VTAM APPLID for a batch region.

## Step 4: Connect the CRM to Back-End Systems on the Mainframe

---

### **Application Program Major Node Name**

This tag identifies the node to the network.

### **TYPE=APPL**

A major node should be an Application Major Node.

### **ACBNAME=LU Name**

Logical unit name. This can be identical to the Application Major Node name. This is the name used as the batch logical unit.

### **APPC=YES**

Defines an APPC logical unit.

### **DLOGMOD=xxx**

This is the VTAM LOGMODE name used for these sessions. An example is SMSNA100, which is the default WebLogic JAM recommends.

### **PARSESS=YES**

Allows multiple sessions between session partners.

### **SECACPT=[NONE | ALREADYV | CONV]**

This parameter sets the level of security and must match a corresponding value for the CRM.

**NONE** means no security credentials are provided with requests sent on the link.

**ALREADYV** means that the application processing the request needs to identify the user making the request for access control to application resources. A user ID is provided with the request.

**CONV** means that full security credentials are required for any requests sent on the link. A user ID and password are required for each request.

### **SYNCLVL=[CONFIRM | SYNCPT]**

Determines level of transactional support. **CONFIRM** supports non-transactional DPL and APPC service requests only, while **SYNCPT** will in addition, support transactional service requests. See [“Administering Transactions”](#) for information on these options.

### **DSESLIM=n**

Defines the total number of sessions for this connection, and the number that are established by the batch region.

### **DMINWNL=n**

Defines the sessions that are established by the APPC/MVS batch region. It is recommended that you set the **WINNER** numbers to be the remainder of the

## 2 Configuring WebLogic JAM Connectivity

---

MAXIMUM SESSIONS after you define the number of WINNER numbers needed by the CRM.

### Step 2: Add a Logical Unit for a Batch Region to APPC/MVS

After you create a batch logical unit (VTAM APPLID), you must add it to APPC/MVS. [Listing 2-10](#) is a sample for adding the batch logical unit to APPC/MVS.

#### **Listing 2-10 Sample Command for Associating the Logical Unit with a Batch Region**

---

```
LUADD ACBNAME(MVSAPPL1) BASE SCHED(ASCH) TPDATA(SYS1.APPCTP)
SIDEINFO DATASET(SYS1.APPCSI)
```

---

The following parameters represent the minimum requirements for adding a logical unit to APPC/MVS.

**ACBNAME( xxx )**

This should match the ACBNAME of the batch logical unit's VTAM APPLID.

**SCHED( xxx )**

This specifies that the APPC/MVS scheduler is the scheduler for all requests on this logical unit.

### WebLogic Administration Console Parameters for Batch Region Connectivity

When you configure WebLogic JAM connectivity for a batch region in the WebLogic Administration Console, you will need to supply the following parameters:

- [Batch Region Logical Unit](#)
- [Batch Region CRM Link](#)

**Note:** A worksheet is available to give you a central location for gathering various connection parameters before you sit down to enter them into the WebLogic Administration Console in [“Step 5: Enter Connectivity Information into WebLogic Administration Console.”](#) For a copy of the worksheet, see [“Mainframe Connectivity Worksheet.”](#)

### Batch Region Logical Unit

[Table 2-7](#) shows the parameters to use in the WebLogic Administration Console to configure a batch region's logical unit. Work closely with your administrator to get the CRM logical unit defined in your SNA network.

**Table 2-7 Batch Region Logical Unit Definition Parameters in the WebLogic Administration Console**

WebLogic Administration Console Parameters	Description	Parameter Syntax
Name	The arbitrary name of the region to which the CRM connects.	Alpha-numeric string Ex. Batch1
Logical Unit	The batch logical unit name. The application identifier of the APPC/MVS logical unit name of the batch region to which the CRM connects.	Eight character alpha-numeric string Ex. MVSAPPL1

### Batch Region CRM Link

[Table 2-8](#) shows the parameters to use in the WebLogic Administration Console to configure batch region's CRM Link. WebLogic JAM uses a CRM Link definition to store the parameters of the connection (or link) between the CRM's logical unit and the logical unit of the back-end system. These parameters determine the characteristics of the logical connection made between the CRM and the batch region.

**Table 2-8 Batch Region CRM Link Definition Parameters in the WebLogic Administration Console**

<b>WebLogic Administration Console Parameters</b>	<b>Description</b>	<b>Parameter Syntax</b>
Name	The arbitrary name of the CRM Link for this batch region.	Alpha-numeric string Ex. CRM1Batch1
Region Name	The name of the batch region for which you are supplying this link.  This should correspond to the arbitrary name provided for the batch region.	Alpha-numeric string Ex. Batch1
Logmode Name	The class of services for this link.  This should be the DLOGMOD on the batch session definition.	Alpha-numeric string Ex. SMSNA100
Maximum Sessions	Defines the total number of sessions for this link.  The value of DSESLIM from the batch logical unit definition.	Numeric value Ex. 20
Minimum Winner Sessions	The number of winner sessions pre-allocated to the CRM for this link.  This value should be less than or equal to the maximum session count minus the value of DMINWNL from the batch logical unit definition.	Numeric value Ex. 10
Security Level	The security credentials required for requests on this link.  This should be compatible with the corresponding value provided in the SECACPT(NONE   ALREADYV   CONV) parameter of the batch connection definition. For more information, see <a href="#">“Step 1: Create a Logical Unit for a Batch Region and Add it to APPC/MVS.”</a>	Supported values available in console list (Local, Identify, Verify). Ex. Local

WebLogic Administration Console Parameters	Description	Parameter Syntax
Default User ID	A user ID to be provided for requests from applications that do not provide security credentials.  This parameter is for use only with Security Level IDENTIFY.	Alpha-numeric string  Ex. USER01

# Step 5: Enter Connectivity Information into WebLogic Administration Console

At this point you and your system administrators should have established all of the connections to your back-end systems using steps 1-3. You are now ready to enter this information into the WebLogic Administration Console so that WebLogic JAM components will connect to your CICS, IMS or batch regions upon startup. Use the following steps to enter connectivity parameters into the WebLogic Administration Console:

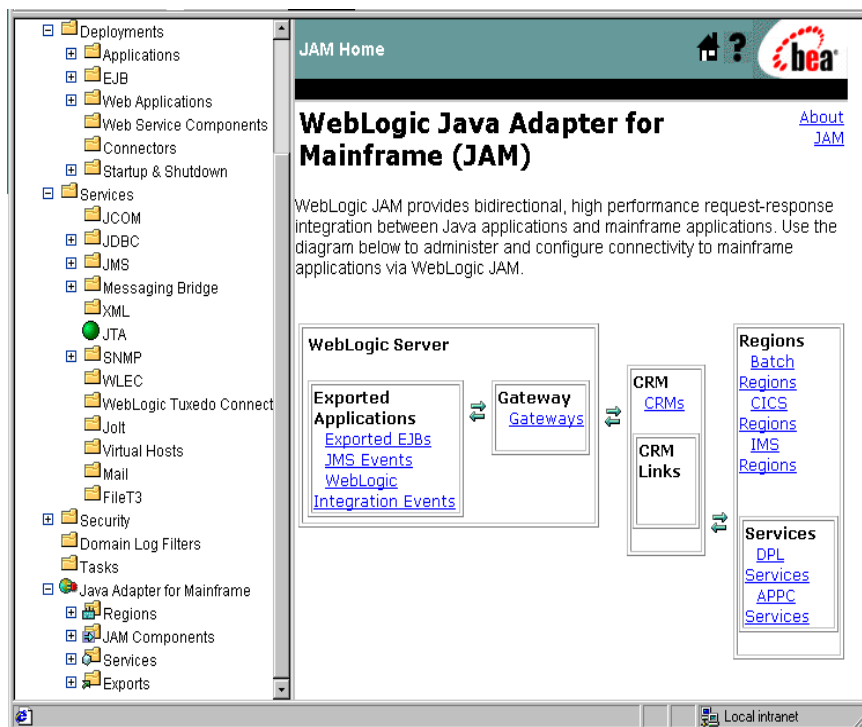
- [Step 5.1: Create Region Definitions](#)
- [Step 5.2: Create CRM Definitions](#)
- [Step 5.3: Create CRM Link Definitions](#)

## Step 5.1: Create Region Definitions

The process for creating and modifying region definitions is similar for all three types of regions. For the purposes of demonstration, the steps in this section show the process of creating a CICS region.

## 2 Configuring WebLogic JAM Connectivity

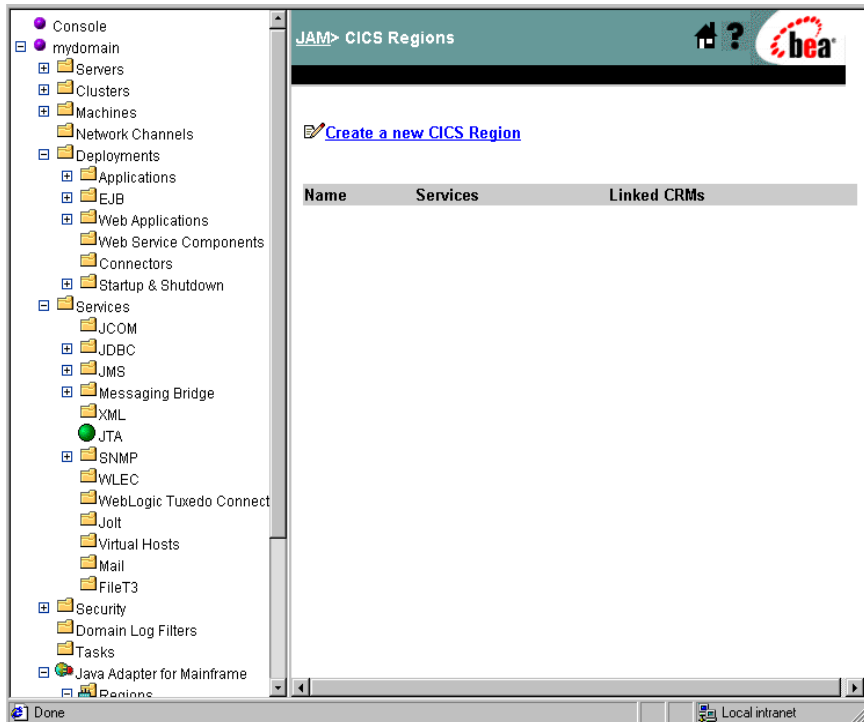
To create a region definition, perform the following steps:





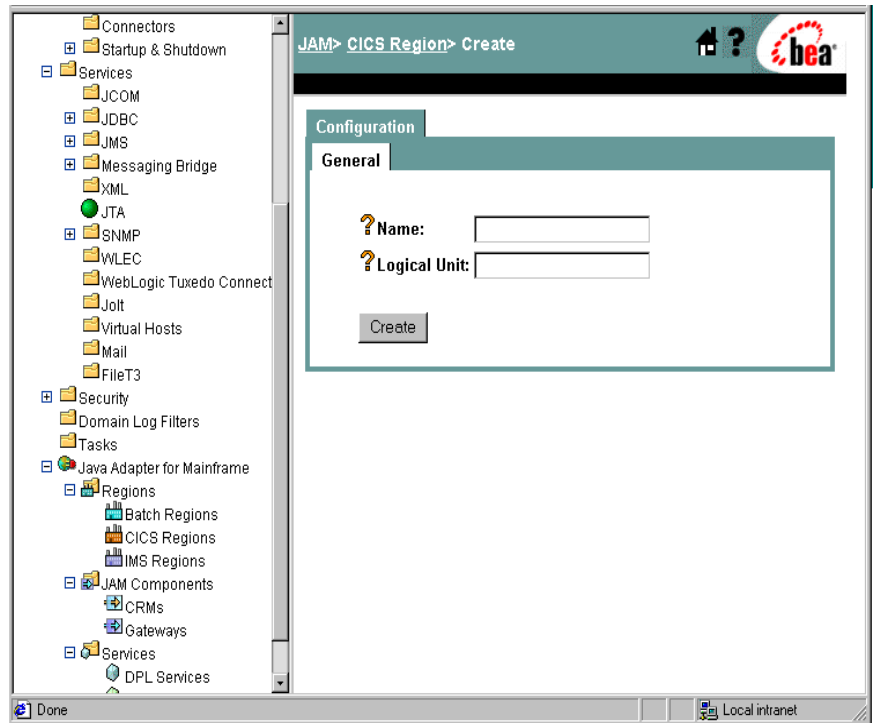
## Step 5: Enter Connectivity Information into WebLogic Administration Console

1. From the WebLogic JAM home page, click on the type of region for which you want to create a definition (batch, CICS, IMS). The List Regions Page displays.



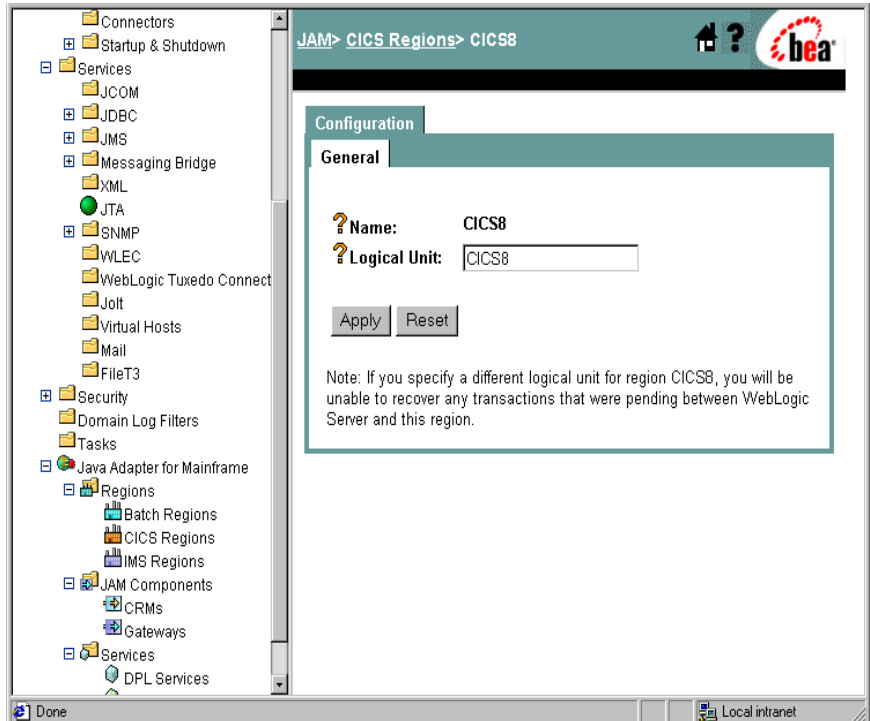
## 2 Configuring WebLogic JAM Connectivity

### 2. Click **Create a new CICS Region**.



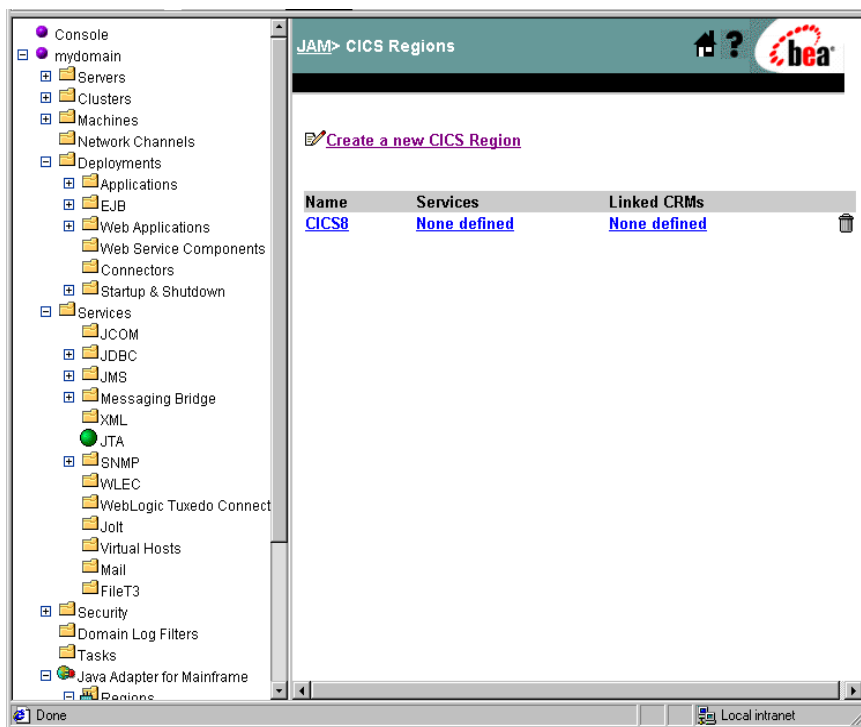
## Step 5: Enter Connectivity Information into WebLogic Administration Console

3. Enter the configuration parameters in the appropriate fields. Click **Create** to submit your parameters and create your CICS region.



## 2 Configuring WebLogic JAM Connectivity

4. Click **CICS Regions** in the top navigation bar to return to the Regions page.



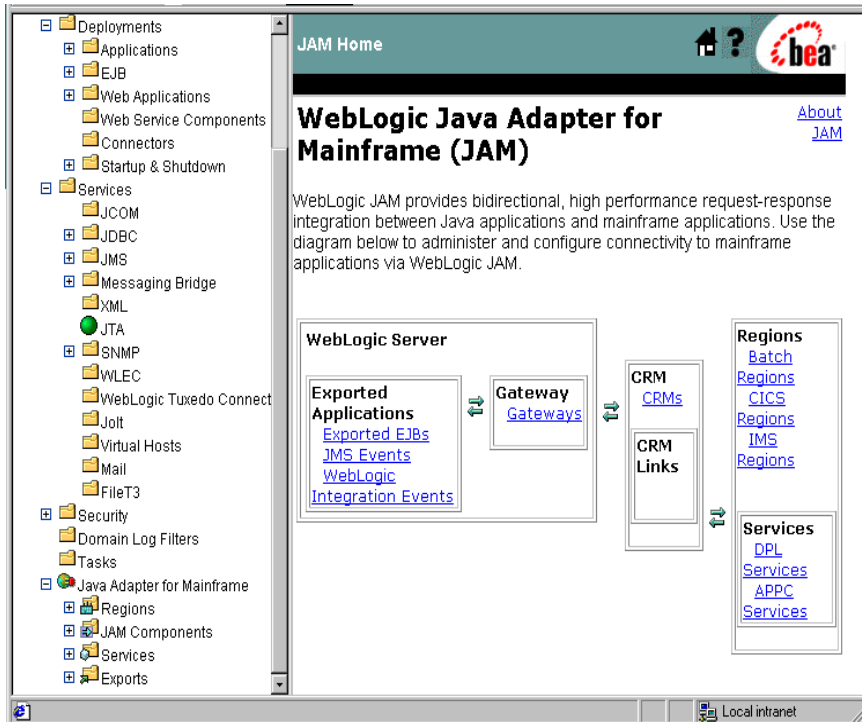
5. Verify that your new region definition appears in the Regions page.

**Note:** The Service and CRM Link parameters for your new region definition are not defined yet. You will have to define these to establish connectivity. Continue to follow the steps in this section to complete the connectivity process; however, for immediate information and instructions for defining a CRM link, see “[Step 5.3: Create CRM Link Definitions.](#)”

Services are set up as part of integrating WebLogic applications with Mainframe applications after you have finished configuring connectivity. For detailed information about application integration with WebLogic JAM, included configuration for services, see “Exposing Mainframe Applications to J2EE Clients.”

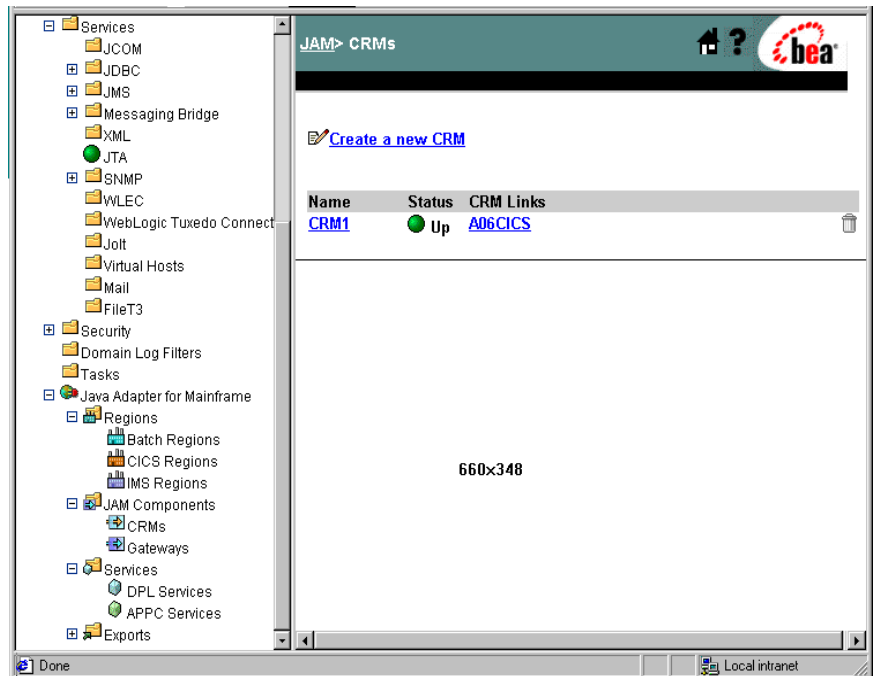
## Step 5.2: Create CRM Definitions

To create a CRM definition, perform the following steps:



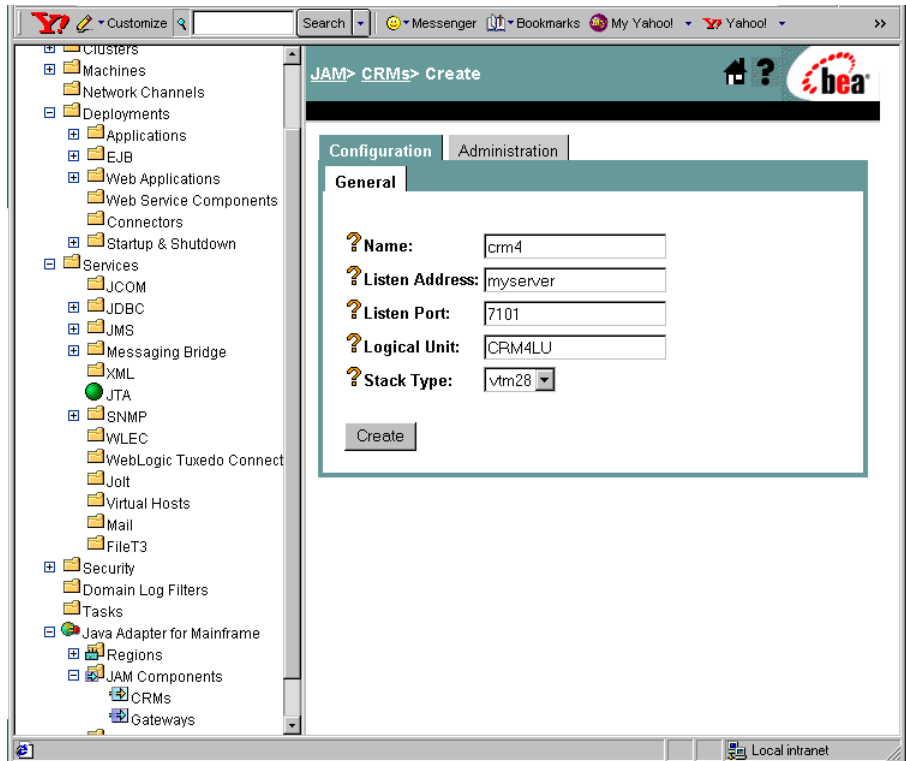
## 2 Configuring WebLogic JAM Connectivity

1. From the WebLogic JAM home page, click **CRMs**. The List CRM page displays.



## Step 5: Enter Connectivity Information into WebLogic Administration Console

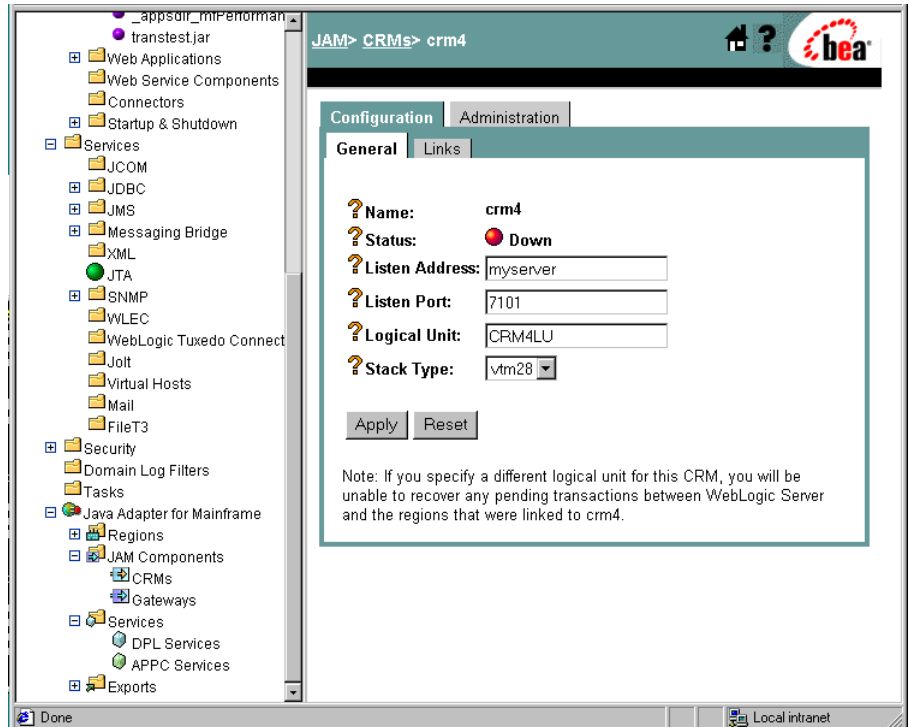
### 2. Click **Create a new CRM**.



## 2 Configuring WebLogic JAM Connectivity

3. Enter the configuration parameters from your Mainframe Connectivity Worksheet in the appropriate fields. Click **Create** to submit your parameters.

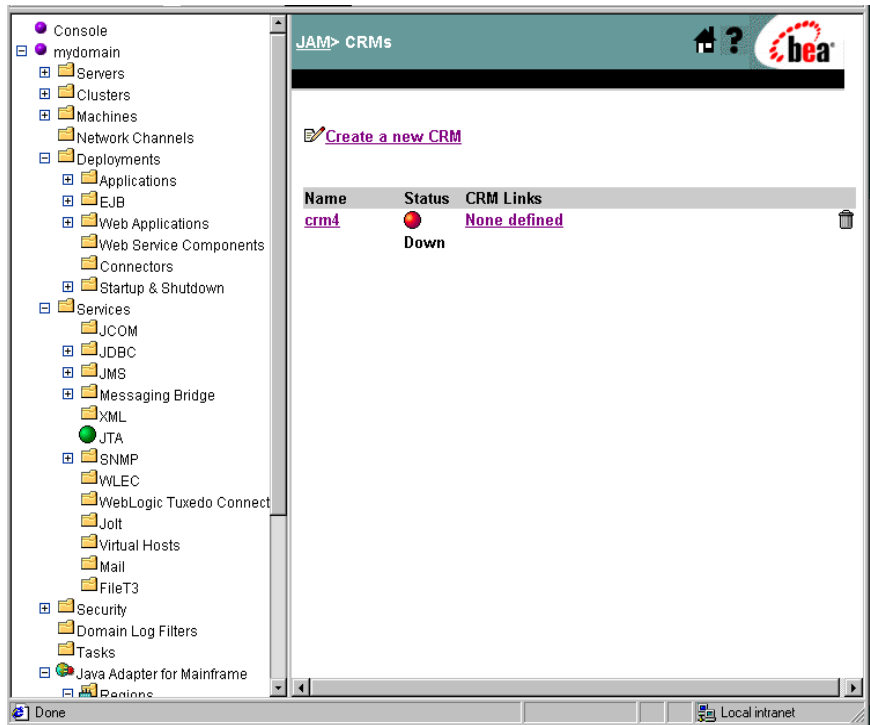
**Note:** Notice that the status of your newly created CRM is down. Continue to follow the steps in this section to complete the connectivity process; however, for information on starting and stopping a CRM, see “Starting the CRM.”





## Step 5: Enter Connectivity Information into WebLogic Administration Console

- Click **CRMs** in the top navigation bar to return to the CRM page.

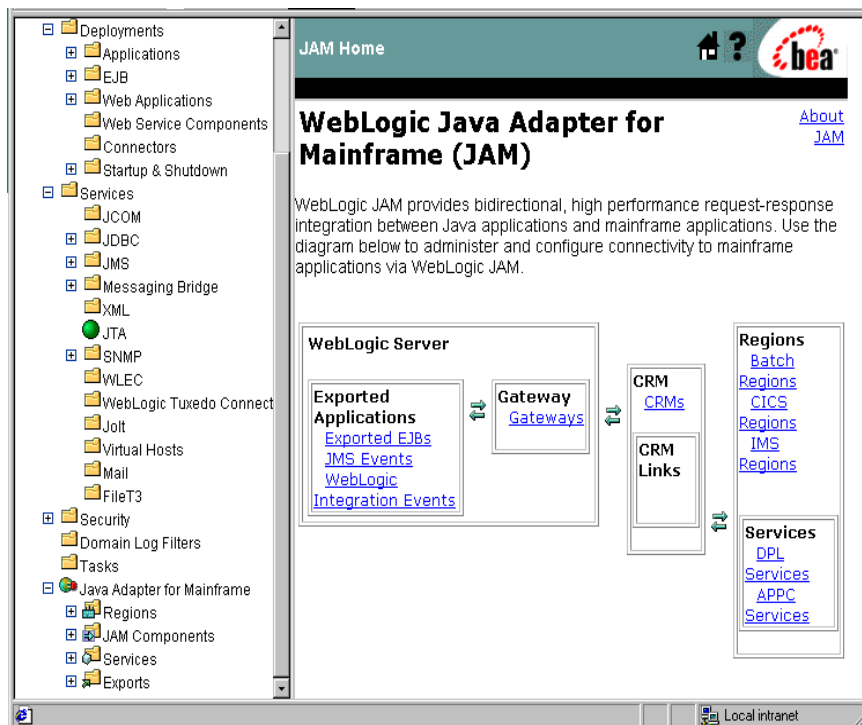


- Verify that your new CRM definition appears in the CRM page.

**Note:** Notice that the CRM Link parameter for your new CRM definition is not defined yet. Continue to follow the steps in this section to complete the connectivity process; however, for immediate information and instructions for defining a CRM link, see “[Step 5.3: Create CRM Link Definitions.](#)”

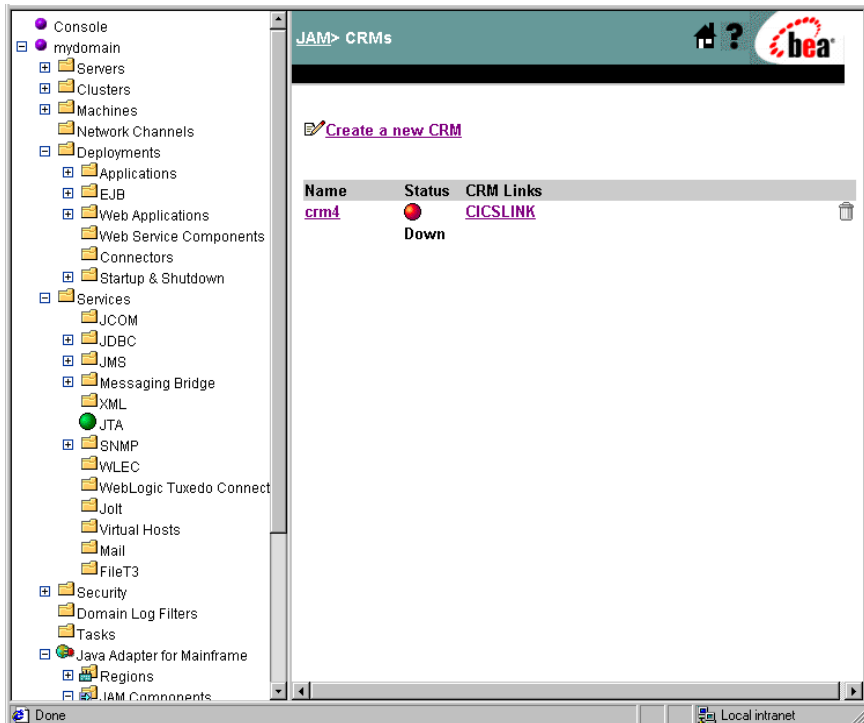
### Step 5.3: Create CRM Link Definitions

To create a CRM Link definition, perform the following steps:



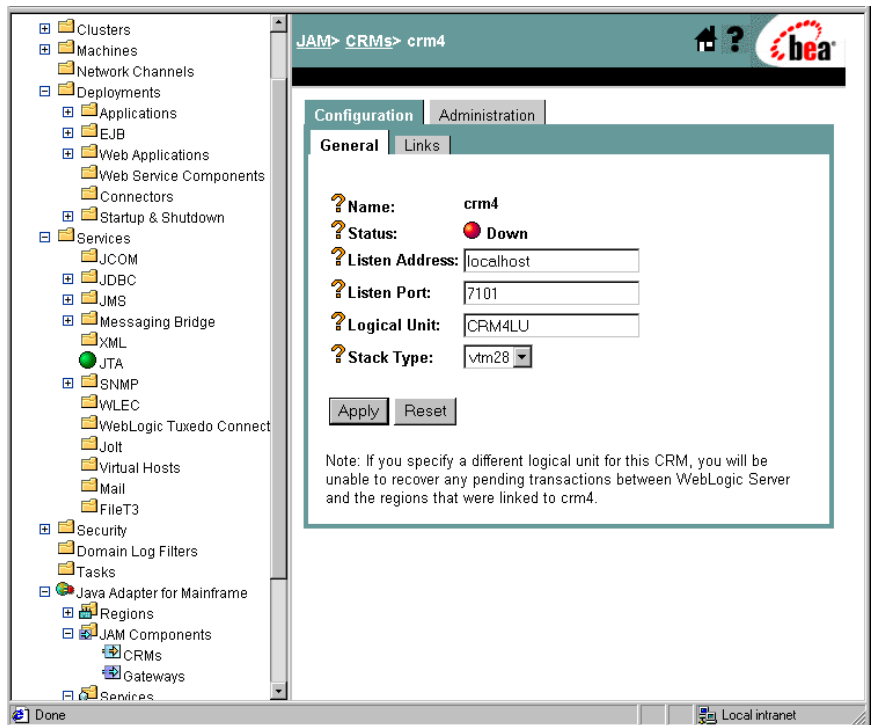
## Step 5: Enter Connectivity Information into WebLogic Administration Console

1. From the WebLogic JAM home page, click **CRMs**. The List CRM page displays.



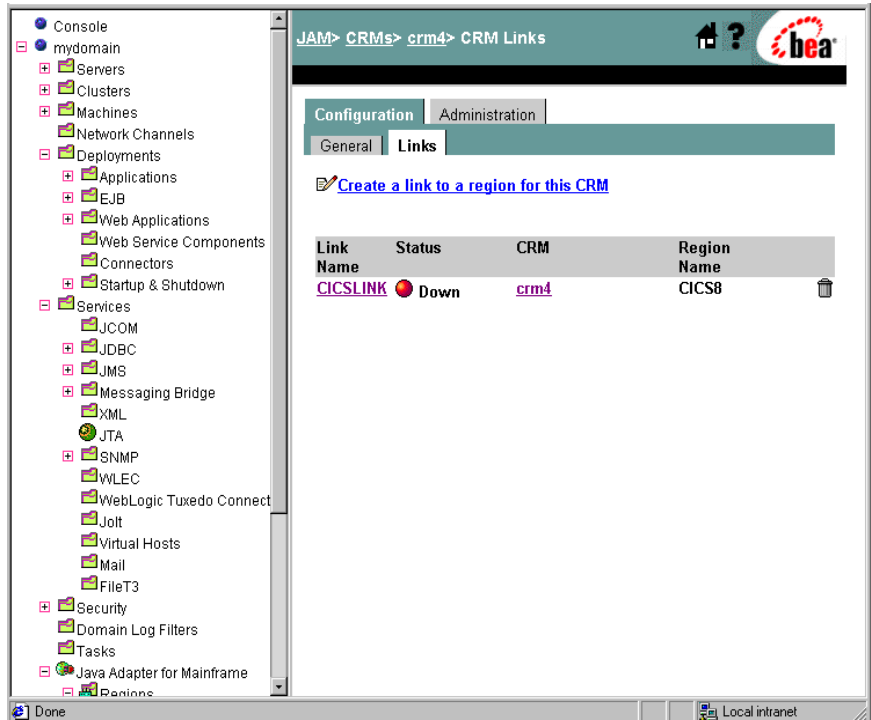
## 2 Configuring WebLogic JAM Connectivity

2. Click the existing CRM for which you want to establish a CRM link.



## Step 5: Enter Connectivity Information into WebLogic Administration Console

3. Click the Links tab. A page listing all of the CRM Links established for the selected CRM displays.



## 2 Configuring WebLogic JAM Connectivity

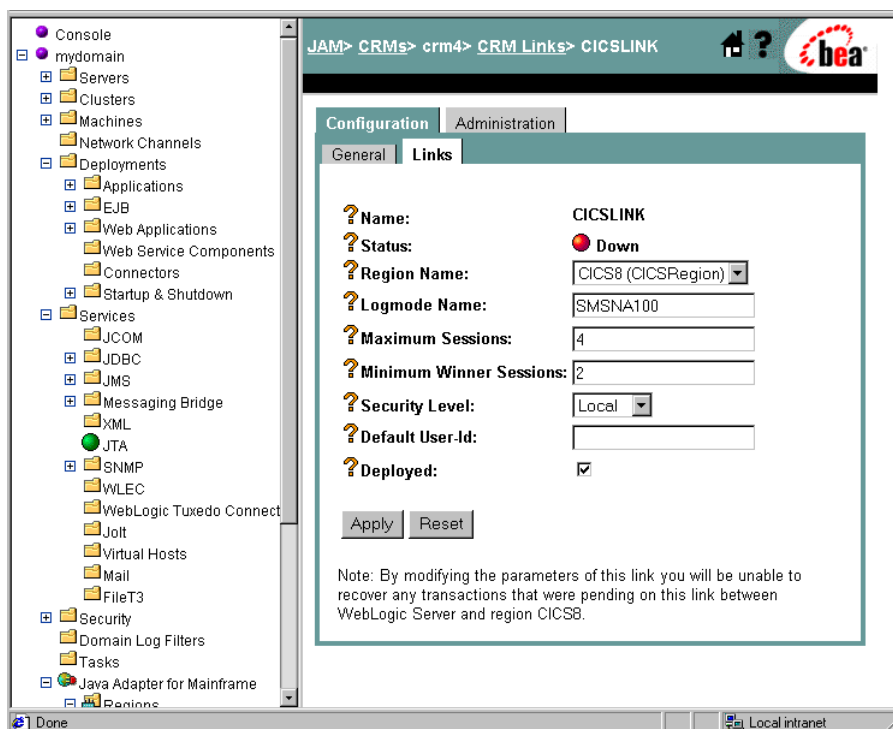
### 4. Click **Create a link to a region for this CRM**.

**Note:** A CRM Link cannot be created unless a region exists that is not linked to this CRM.

The screenshot shows the BEA WebLogic JAM Configuration console. On the left is a tree view of the configuration hierarchy, including Console, mydomain, Servers, Clusters, Machines, Network Channels, Deployments, Applications, EJB, Web Applications, Web Service Components, Connectors, Startup & Shutdown, Services, JCOM, JDBC, JMS, Messaging Bridge, XML, JTA, SNMP, WLEC, WebLogic Tuxedo Connect, Jolt, Virtual Hosts, Mail, FileT3, Security, Domain Log Filters, Tasks, Java Adapter for Mainframe, and Regions. The main panel displays the 'JAM> CRMs> crm4> Create' dialog. The 'General' tab is active, showing fields for Name, Region Name (set to CICS8 (CICSRegion)), Logmode Name (set to SMSNA100), Maximum Sessions (set to 4), Minimum Winner Sessions (set to 2), Security Level (set to Local), Default User-Id, and a Deployed checkbox which is checked. A 'Create' button is at the bottom of the dialog. The status bar at the bottom indicates 'Done' and 'Local intranet'.

### 5. Enter the configuration parameters in the appropriate fields. Click the **Deployed** check box if you want the link to be configured and usable at startup. Click **Create** to submit your parameters.

**Note:** Notice that the status of your newly created CRM Link is down. Continue to follow the steps in this section to complete the connectivity process.



## Step 6: Define a WebLogic JAM Gateway

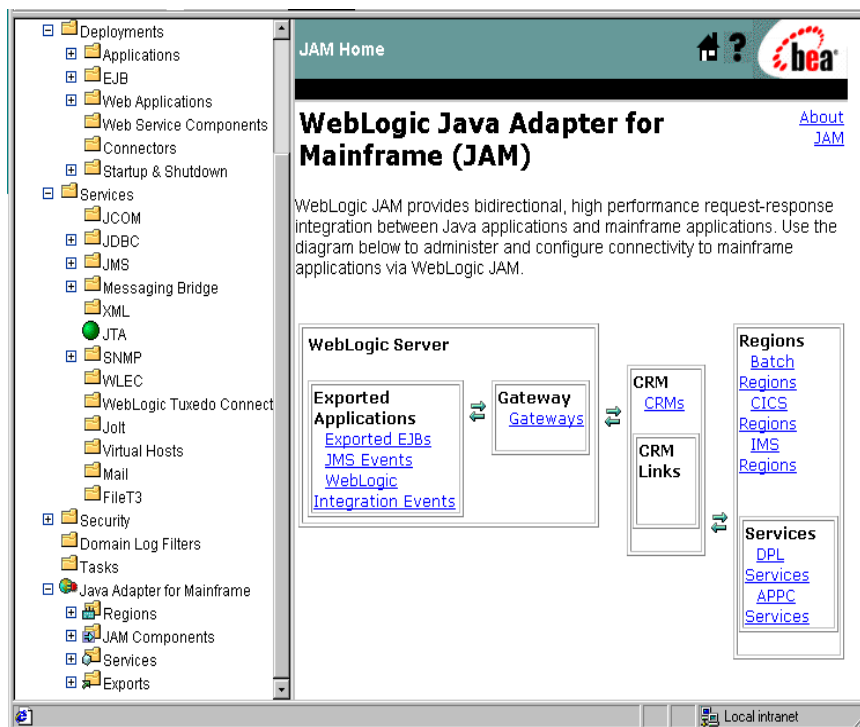
Once you have created definitions for the CRM, CRM Link, and region information, you need to define a WebLogic JAM Gateway. When you define a WebLogic JAM Gateway you define where the Gateway runs in your WebLogic domain, and to which CRM it will connect.

## 2 Configuring WebLogic JAM Connectivity

A WebLogic JAM Gateway requires a CRM; therefore, the Gateway cannot be created until a CRM exists. Also, the Gateway requires an instance of WebLogic Server that does not already have a Gateway defined for it.

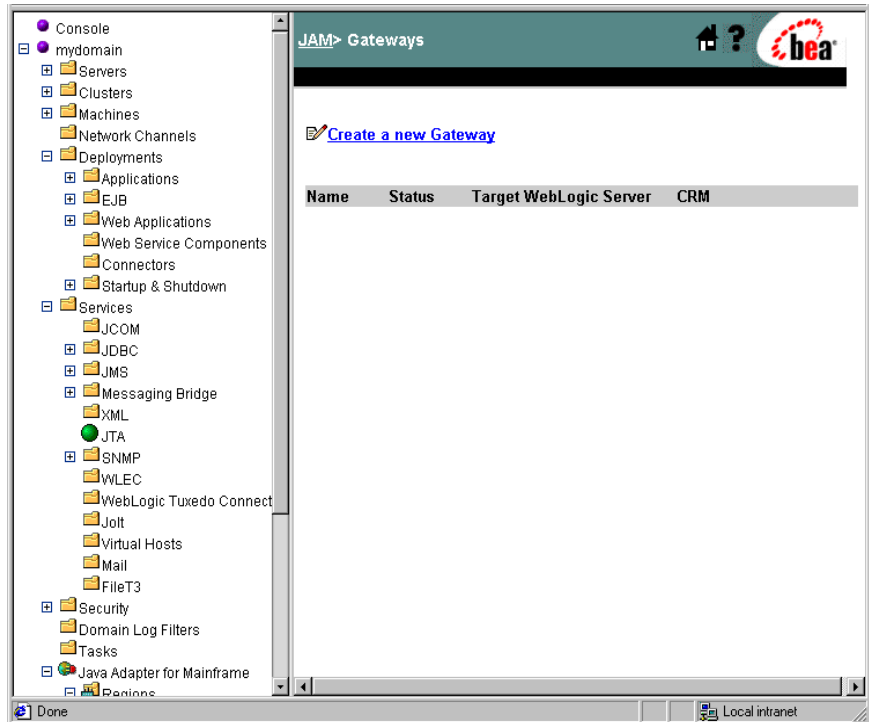
A WebLogic JAM Gateway can run on any instance of WebLogic Server within your WebLogic domain. However, you can only have one Gateway defined for each instance of WebLogic Server. Multiple WebLogic JAM Gateways can be defined to connect to the same CRM, to enable features such as load balancing and failover. For detailed information on using WebLogic JAM in a multi-Gateway environment, see [“Connecting Multiple Gateways to a Single CRM.”](#)

To define a Gateway, follow these steps:



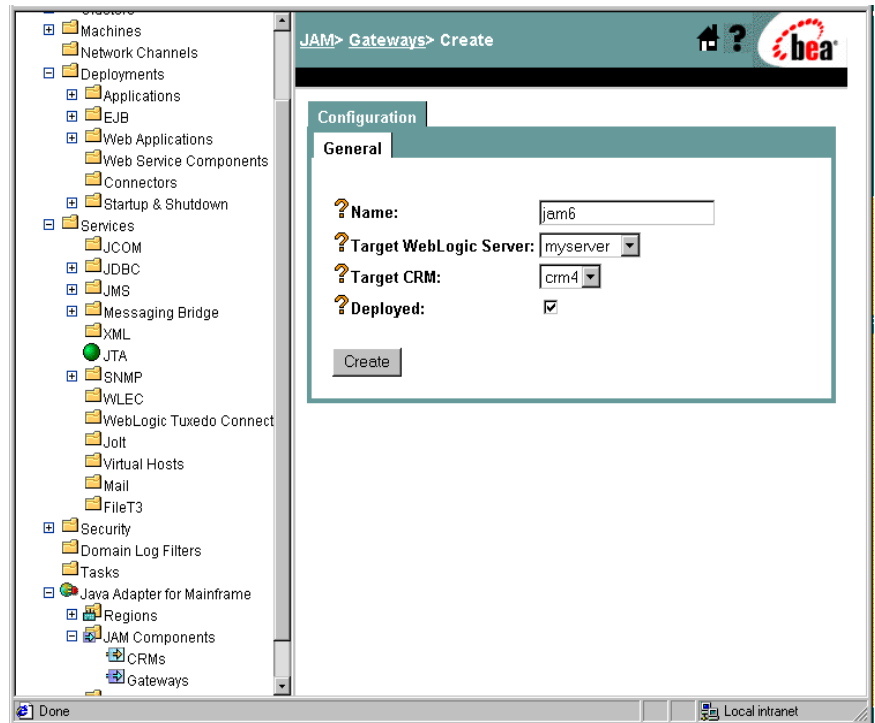


1. From the WebLogic JAM home page, click **Gateways**. The List Gateways page displays.



## 2 Configuring WebLogic JAM Connectivity

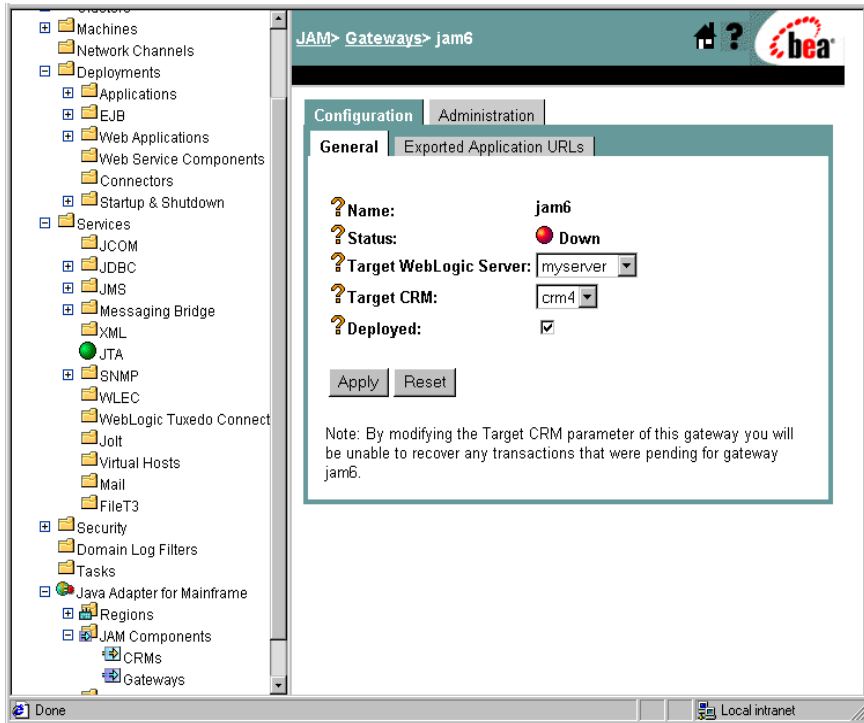
2. Click **Create a new Gateway**.



3. Enter the configuration parameters in the appropriate fields. Click the **Deployed** check box so it is checked.

- Click **Create** to submit your parameters.

**Note:** A Gateway can be marked as active or inactive by selecting (or deselecting) the Deployed check box.



**Note:** Notice that the status of your newly created Gateway is down. Continue to follow the steps in this section to complete the connectivity process; however, for information on starting and stopping a Gateway, see [“Starting and Stopping a Gateway.”](#)

# Step 7: Verify Your WebLogic JAM Connectivity Configuration

Now that connectivity has been configured, you may verify your configuration using the following steps:

1. Start the CRM. For detailed information and steps for starting the CRM, see [“Starting the CRM.”](#)
2. Start the Gateway. For detailed information and steps for starting the Gateway, see [“Starting and Stopping a Gateway.”](#)
3. Check CRM Link status on the List CRM Links page to see that it came up. Its status should now be up. For detailed information and steps for checking CRM links, see [“Modifying Your Configuration.”](#)

# 3 Integrating Applications With WebLogic JAM

This section provides information and steps to help guide you through the process of determining which applications you want to integrate, capturing the parameters necessary for configuration in the WebLogic Administration Console, and configuring your mainframe client applications.

This section provides information on the following subjects:

- [Understanding Application Integration with WebLogic JAM](#)
- [Exposing Mainframe Applications to J2EE Clients](#)
- [Exposing J2EE Applications to Mainframe Clients](#)

## Understanding Application Integration with WebLogic JAM

WebLogic JAM enables you to easily reuse existing mainframe applications and integrate them with new applications built on WebLogic Server. CICS programs, IMS programs, and APPC programs can be integrated with WebLogic applications without

changes to the original mainframe application. Existing mainframe applications can be exposed to WebLogic applications with configuration using the WebLogic Administration Console.

## Application Integration Considerations

After defining the connectivity of the CRM to the back-end application environment, you must identify applications that you want to access. Accessing these applications include determining the link and the region where the mainframe application resides. It also includes determining how to handle data translation.

Discuss the following list of topics with your system administrator. These topics are relevant to both the WebLogic and mainframe application environments.

- [Data Translation](#)
- [Access to Mainframe and WebLogic Applications](#)

### Data Translation

It is usually necessary to convert the format of the data passed between WebLogic Server applications and mainframe applications. It may require collecting information about data formats from a mainframe application (usually a copybook) for configuration of these applications. For additional information about using data translation in your applications, see the *BEA WebLogic Java Adapter for Mainframe Programming Guide*.

For information about using schemas to perform data translation with WebLogic JAM and WebLogic Integration, refer to the *BEA WebLogic Java Adapter for Mainframe Workflow Processing Guide*.

### Access to Mainframe and WebLogic Applications

In considering application integration, you should determine whether you need to access WebLogic applications from the mainframe or access mainframe applications from WebLogic.

- Access mainframe applications from WebLogic

When you create a service definition to expose a mainframe application to a WebLogic application, a service entry in the WebLogic Administration Console maps a service name to the name of the application on the mainframe. It also defines various other parameters depending on the type of mainframe application. You must also determine the mainframe region where the application resides as well as the link used by the CRM to access that region.

When a WebLogic application sends a request to WebLogic JAM with this service name, WebLogic JAM invokes the mainframe application that is mapped to the specified service name.

- Access WebLogic applications from the mainframe

Applications built on WebLogic can also be invoked by mainframe applications via WebLogic JAM. By creating an export definition within the WebLogic Administration Console, you can configure mainframe applications to invoke EJBs, enqueue data to a JMS queue, or enqueue data to the workflow engine of WebLogic Integration. The export definition maps a service name, specified by the mainframe application, to a WebLogic application. When a mainframe application sends a request to WebLogic JAM with this service name, WebLogic JAM invokes the configured WebLogic application.

# Exposing Mainframe Applications to J2EE Clients

WebLogic JAM enables CICS, standard IMS, and APPC/MVS applications (that run in CICS, IMS, or as a batch job) to be invoked from the WebLogic environment.

This section provides information on the following subjects:

- [Exposing a CICS Distributed Program Link as a DPL Service](#)
- [Exposing a Standard IMS Transaction Program as an APPC Service](#)
- [Exposing an APPC/MVS Transaction Program as an APPC Service](#)

## Exposing a CICS Distributed Program Link as a DPL Service

Distributed Program Link (DPL) programs are CICS programs that accept a `COMMAREA` (short for communication area) as input. To configure WebLogic JAM to invoke a CICS DPL program, you must enter configuration parameters into the WebLogic Administration Console.

This section provides information about the following:

- [Steps for Exposing a DPL Program](#)
- [WebLogic Administration Console Parameters for DPL Services](#)

You will need certain parameter information to complete WebLogic JAM configuration, such as program name and WebLogic service name. You will also need to select the region in which the service is available. This section provides information to help you understand and capture the parameters you will need to enter into the WebLogic Administration Console in a later step.

### Steps for Exposing a DPL Program

To configure WebLogic JAM to invoke a CICS DPL program, you must determine the data translation considerations and the program definition for the CICS region.

- [Step 1: Determine Data Translation Requirements](#)
- [Step 2: Determine Program Access](#)

#### Step 1: Determine Data Translation Requirements

The data format of the DPL program is given by the declaration of the `COMMAREA` (short for communication area). For a DPL program written in COBOL, the `COMMAREA` is declared in the `LINKAGE` section of the program. You may need to obtain the `COMMAREA` copybook used in the `LINKAGE` section for use with WebLogic JAM's eGen Application Generator or WebLogic Integration. For information about obtaining and using the copybook declaration, refer to the *BEA WebLogic Java Adapter for Mainframe Programming Guide*.



For information about using schemas to perform data translation with WebLogic JAM and WebLogic Integration, refer to the *BEA WebLogic Java Adapter for Mainframe Workflow Processing Guide*.

### Step 2: Determine Program Access

In order to access a CICS program with a DPL request, there must be a CICS program resource definition that defines the CICS program to the CICS region. If a resource definition does not exist, ask your CICS administrator to create one.

A distributed program link occurs when the program is invoked on a “different” CICS system from the invoker. A special CICS program (DFHMIRS) handles the distributed request and invokes the designated CICS program. When the DFHMIRS program invokes the CICS program, the CICS program becomes associated with either the mirror transaction ID `CSMI` or `CVMI`, or an alternate transaction for DFHMIRS (as defined by the system administrator.) Transaction IDs are used in CICS for resource tracking and to restrict access to database resources.

It may be advantageous to use alternate transaction IDs to execute the DPL program so that resources can be tracked and database resources can be restricted. If this DPL program should be associated with a transaction ID other than `CSMI` or `CVMI`, then you must get the transaction ID from your CICS administrator.

### WebLogic Administration Console Parameters for DPL Services

WebLogic JAM requires the parameters shown in Table 3-1 to expose a CICS DPL program link as a DPL service.

**Table 3-1 DPL Service Parameters**

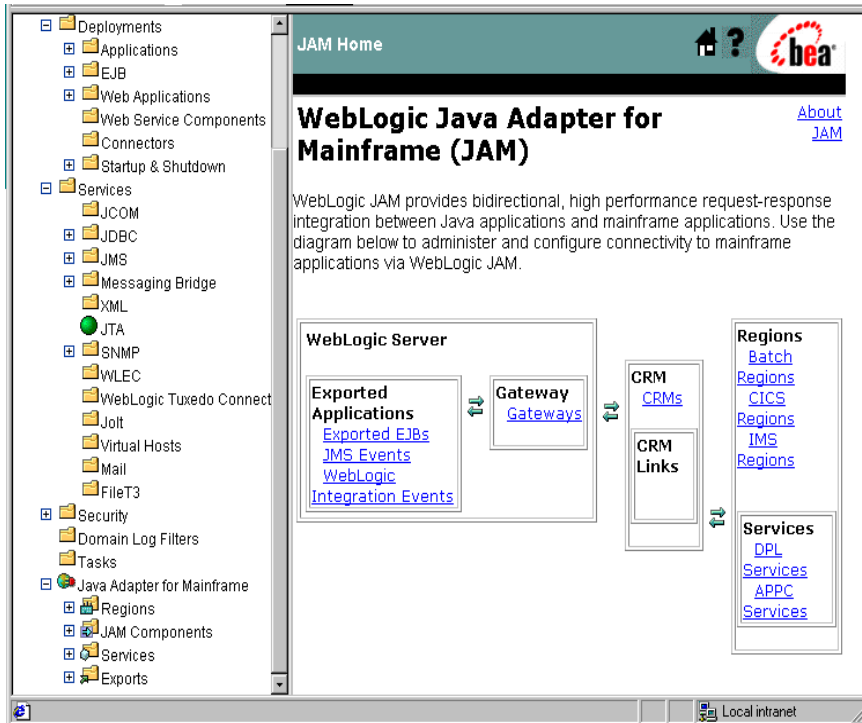
WebLogic Administration Console Parameter	Description	Parameter Syntax
<b>Service Name</b>	An arbitrary name that WebLogic Server applications use to identify this service.	Alpha-numeric string Ex. toupper

WebLogic Administration Console Parameter	Description	Parameter Syntax
<b>CICS Program Name</b>	The name of the CICS program as defined in CICS.  In CICS a program resource definition must exist that defines the DPL program to the CICS region. If one does not exist, ask your CICS administrator to create one.	8 character alpha-numeric string  Ex. TOUPPER
<b>Alternate Transaction ID (if any):</b>	An alternate transaction ID (to replace CSMI or CVMI) used to execute the DPL program. This parameter is optional.	1-4 alpha character  Ex. CXYZ
<b>Timeout:</b>	Approximate amount of time that the program usually takes to complete successfully. WebLogic JAM uses this timeout value to return a timeout error to the calling WebLogic Server application.	Numeric value in milliseconds  Ex. 250
<b>Input and Output Schema (if using WebLogic Integration)</b>	The schema names for this DPL program, if using WebLogic Integration.  <b>Note:</b> For information and instructions for creating an input schema with WebLogic Integration, see the <i>BEA WebLogic Java Adapter for Mainframe Workflow Processing Guide</i> .	Alpha-numeric string  Ex. emprec

#### Steps for Configuring DPL Services in the WebLogic Administration Console

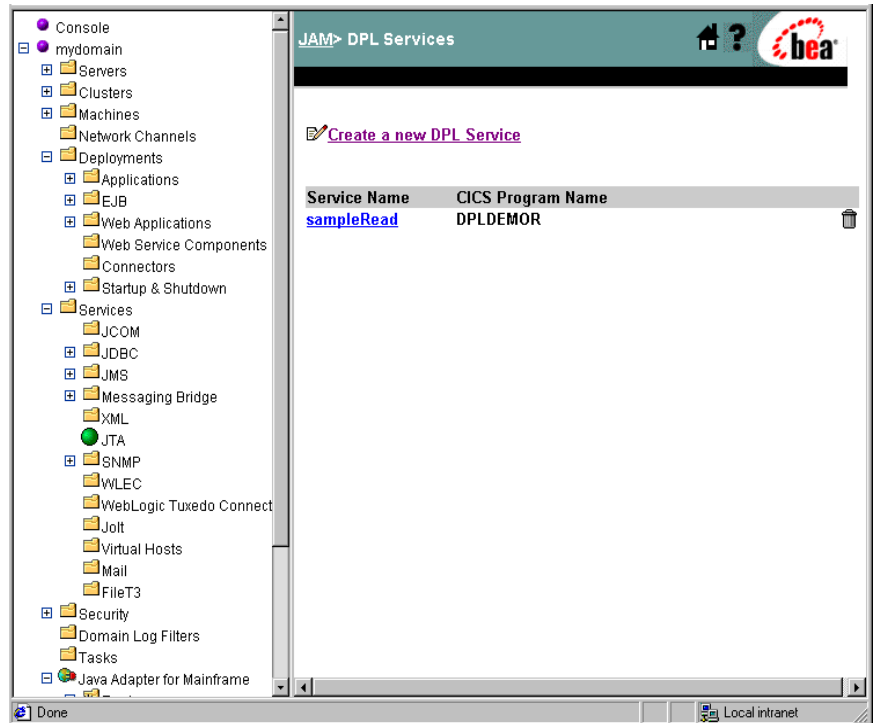
To create the configuration information that WebLogic JAM needs to invoke this DPL program, use the WebLogic Administration Console to perform the following steps:

1. From the WebLogic JAM Home page, click **DPL Services**.

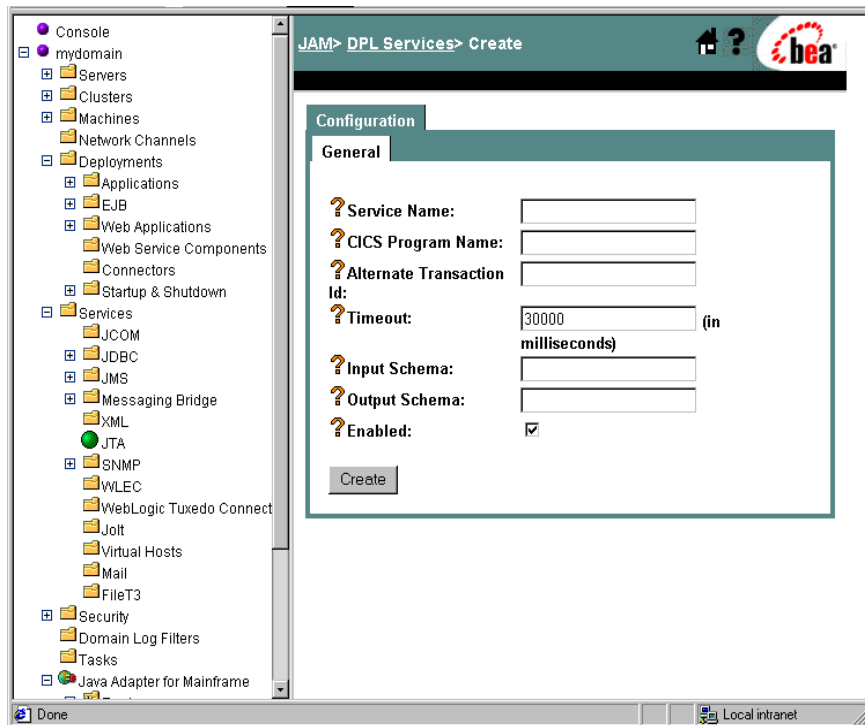


### 3 Integrating Applications With WebLogic JAM

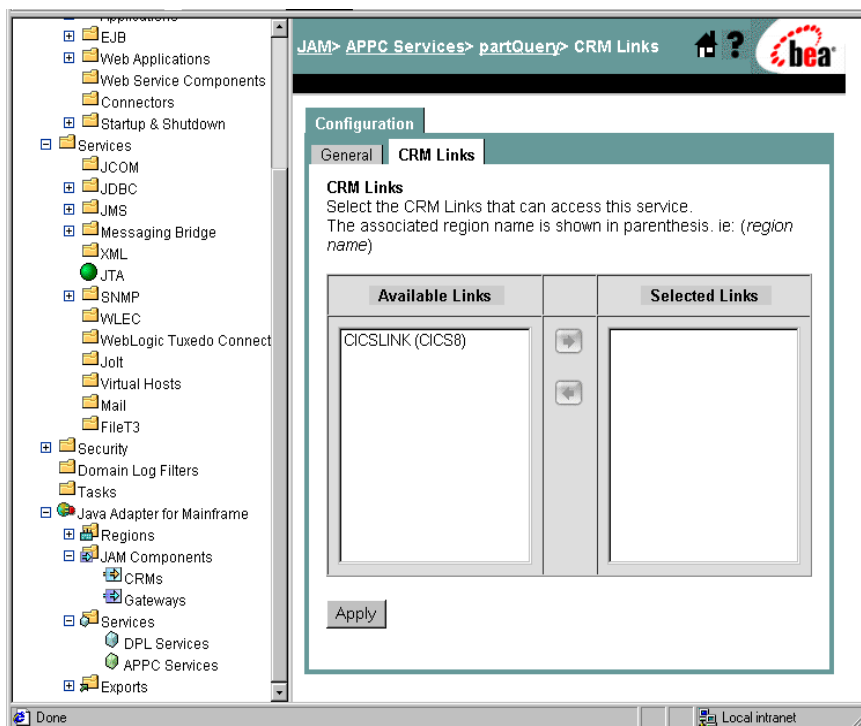
The List DPL Services page displays.



### 2. Click **Create a New DPL Service**.



3. Enter the configuration parameters in the appropriate fields. A check in the **Enabled** box makes the service available for use. Removing the check from the **Enabled** box disables the service. Click **Create** to submit your parameters.



4. Once you have created the DPL service definition in the WebLogic Administration Console, you must select links to the CICS region where the DPL program resides. If the program is available in more than one mainframe region, you may select some or all of the regions in which it is present.

The left pane lists the CICS regions that are defined in the WebLogic JAM configuration. The right pane lists the CICS regions where the DPL program resides. Select Links from the list of all defined CICS regions and move them to the list of CICS regions where this DPL program is available.

5. Click Apply when you have finished adding available links.

## Exposing a Standard IMS Transaction Program as an APPC Service

Standard IMS applications use the DL/I application-programming interface to get input messages and insert output messages. IMS provides implicit APPC support that translates these DL/I calls into APPC communications. WebLogic JAM can invoke these IMS transactions via APPC if the transaction ID of the IMS application is configured. To make this work, you must configure WebLogic JAM with the IMS transaction ID of the program and the IMS regions in which this transaction is available.

This section provides information about the following:

- [Steps for Exposing a Standard IMS Transaction Program](#)
- [WebLogic Administration Console Parameters for an IMS Transaction Program Exposed as an APPC Service](#)

You will need certain parameter information to complete WebLogic JAM configuration. This section provides information to help you understand and capture the parameters you will need to enter into the WebLogic Administration Console in a later step.

### Steps for Exposing a Standard IMS Transaction Program

To configure WebLogic JAM to invoke a standard IMS transaction program, you must determine the data translation considerations and the program definition for the IMS region.

- [Step 1: Determine Data Translation Requirements](#)
- [Step 2: Determine Program Access](#)

Use the following steps to configure WebLogic JAM to make an IMS transaction program available to WebLogic Server applications:

#### Step 1: Determine Data Translation Requirements

The data format of the IMS transaction program is given by the declaration of the Input/Output Program Control Block (IOPCB). When this program is written in COBOL, the IOPCB is declared in the `LINKAGE` section of the program. For

information about obtaining and using the IOPCB declaration copybook from the LINKAGE section, refer to the *BEA WebLogic Java Adapter for Mainframe Programming Guide*.

For information about using schemas to perform data translation with WebLogic JAM and WebLogic Integration, refer to the *BEA WebLogic Java Adapter for Mainframe Workflow Processing Guide*.

#### Step 2: Determine Program Access

In order to access an IMS application, there must be a transaction ID associated with the IMS program. Obtain this transaction ID and IMS region where the transaction program resides from your IMS administrator.

### WebLogic Administration Console Parameters for an IMS Transaction Program Exposed as an APPC Service

WebLogic JAM requires the parameters shown in [Table 3-2](#) to expose an IMS program as an APPC service.

**Table 3-2 IMS Service Parameters**

WebLogic Administration Console Parameter	Description	Parameter Syntax
<b>Service Name</b>	An arbitrary name that WebLogic Server applications use to identify this service.	Alpha-numeric string Ex. toupper
<b>Transaction Program ID</b>	The transaction ID associated with the IMS program.	8 character alpha-numeric string. First character must be an alpha character. Ex. TOUPPER



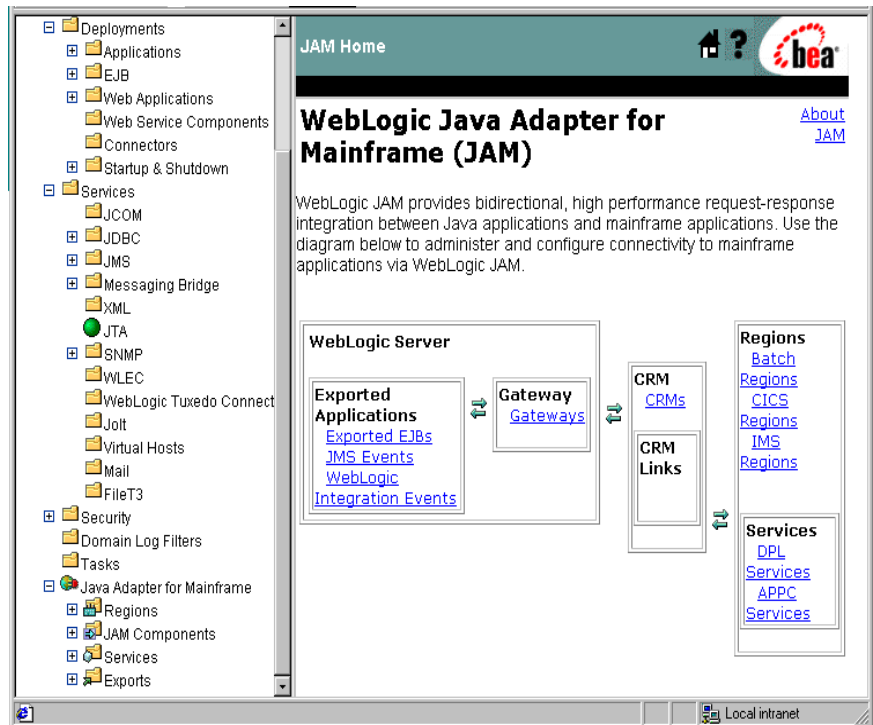
WebLogic Administration Console Parameter	Description	Parameter Syntax
<b>Timeout:</b>	Approximate amount of time that the program usually takes to complete successfully. WebLogic JAM uses this timeout value to return a timeout error to the calling WebLogic Server application.	Numeric value in milliseconds Ex. 250
<b>Input and Output Schema (if using WebLogic Integration)</b>	<b>Note:</b> The schema names for this IMS program, if using WebLogic Integration. For information and instructions for creating an input schema with WebLogic Integration, see the <i>BEA WebLogic Java Adapter for Mainframe Workflow Processing Guide</i> .	Alpha-numeric string Ex. emprec

### Steps for Configuring IMS Programs as APPC Services

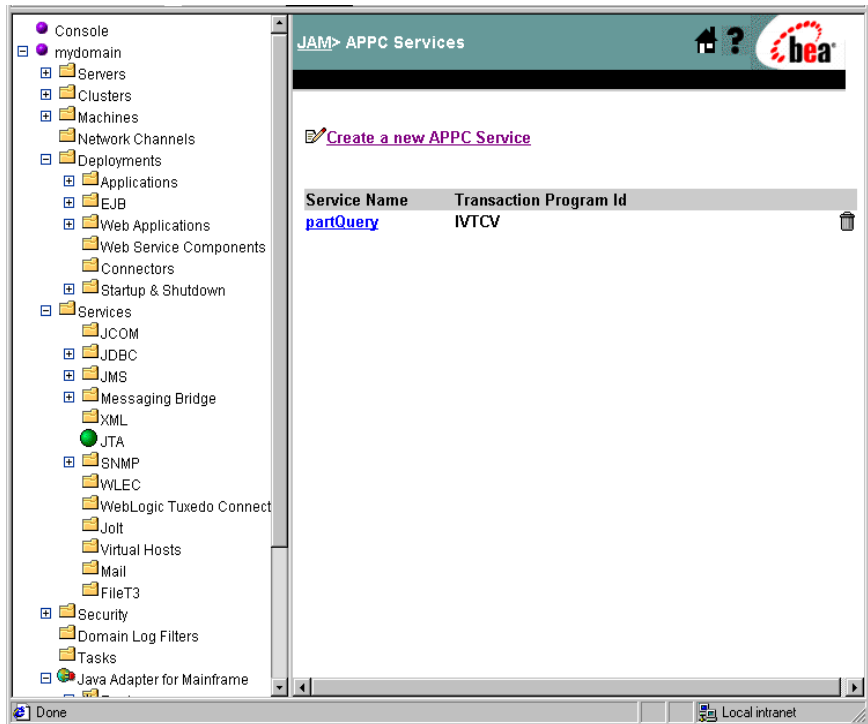
To create the configuration information that WebLogic JAM needs to invoke this IMS transaction program, use the WebLogic Administration Console.

### 3 Integrating Applications With WebLogic JAM

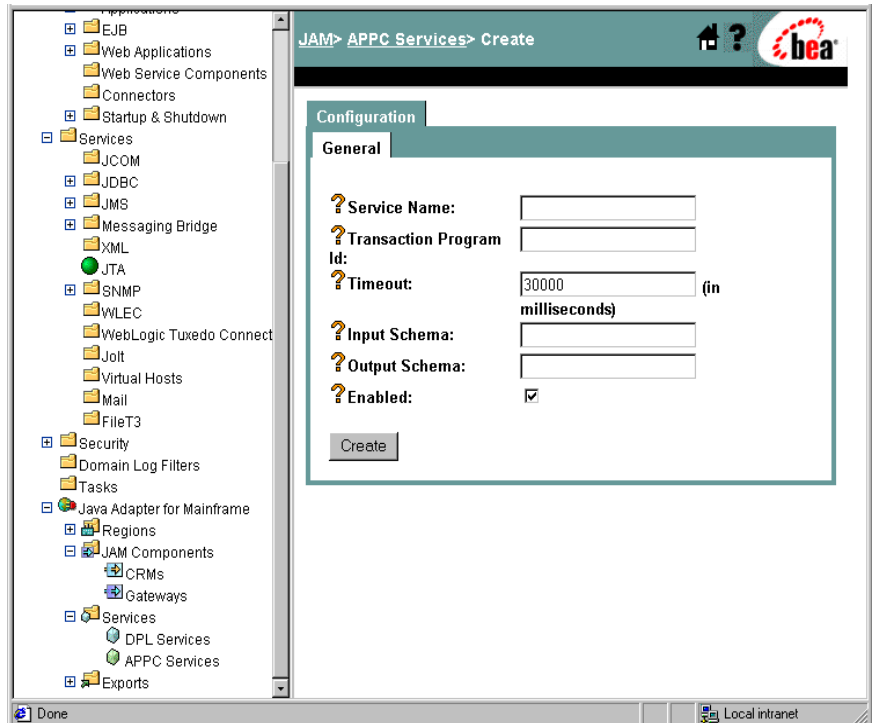
1. From the WebLogic JAM Home page, click **APPC Services**.



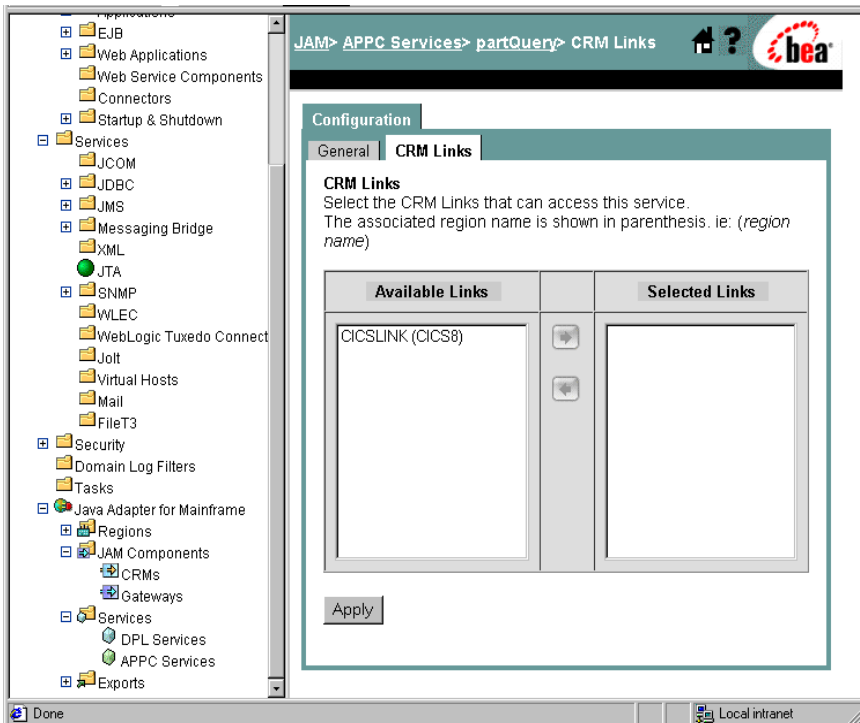
The List APPC Services page displays.



#### 2. Click **Create a New APPC Service**.



3. Enter the configuration parameters in the appropriate fields. A check in the **Enabled** box makes the service available for use when the WebLogic JAM Gateway is started. Removing the check from the **Enabled** box disables the service. Click **Create** to submit your parameters.



4. Once you have created the APPC service definition in the WebLogic Administration Console, you must select a link to the IMS region where the IMS program resides. If the program is available in more than one mainframe region, you may select links to some or all of the regions in which it is present.

The left pane lists the regions that are defined in the WebLogic JAM configuration. The right pane lists regions where this IMS program resides. Select the links from the list of all defined regions and move them to the list of regions where this IMS program is available.

5. Click Apply when you have finished adding IMS regions.

## Exposing an APPC/MVS Transaction Program as an APPC Service

An APPC application is an APPC/MVS batch job, CICS transaction, or IMS transaction that uses the CPI Communications API (CPI-C), or other specific APPC mainframe API, to explicitly communicate with other applications through an APPC conversation. Other examples of mainframe APIs include CICS Distributed Transaction Program (DTP) and the APPC/MVS API. These types of APPC applications may run as batch jobs, CICS transactions, or IMS transactions. You can configure WebLogic JAM to invoke the application through standard APPC protocol regardless of the environment in which the APPC application executes.

This section provides information about the following:

- [Steps for Exposing an APPC/MVS Transaction Program as an APPC Service](#)
- [WebLogic Administration Console Parameters for an APPC Transaction Program Exposed as an APPC Service](#)

You will need certain parameter information to complete WebLogic JAM configuration. This section provides information to help you understand and capture the parameters you will need to enter into the WebLogic Administration Console in a later step.

### Steps for Exposing an APPC/MVS Transaction Program as an APPC Service

To configure WebLogic JAM to invoke an APPC/MVS application as an APPC service, you must determine the data translation considerations and the program definition for the batch region.

- [Step 1: Determine Data Translation Requirements](#)
- [Step 2: Determine Program Access](#)

Use the following steps to configure WebLogic JAM to make an APPC transaction program available to WebLogic Server applications:

### Step 1: Determine Data Translation Requirements

Use WebLogic JAM's eGen utility to generate Java classes (DataViews) that can translate data to and from the mainframe format expected by this APPC program. For programs that are written in COBOL, the data is passed into the `LINKAGE` section. For information about obtaining and using the `LINKAGE` section of the data format of the request and response messages of the APPC program, refer to the *BEA WebLogic Java Adapter for Mainframe Programming Guide*.

For information about using schemas to perform data translation with WebLogic JAM and WebLogic Integration, refer to the *BEA WebLogic Java Adapter for Mainframe Workflow Processing Guide*.

### Step 2: Determine Program Access

In order to access an APPC/MVS application, there must be a transaction profile definition associated with the application. Obtain this transaction profile name where the transaction program resides from your system administrator.

## WebLogic Administration Console Parameters for an APPC Transaction Program Exposed as an APPC Service

WebLogic JAM requires the parameters shown in Table 3-3 to expose an APPC/MVS program as an APPC service.

**Table 3-3 APPC Service Parameters**

WebLogic Administration Console Parameter	Description	Parameter Syntax
<b>Service Name</b>	An arbitrary name that WebLogic Server applications use to identify this service.	Alpha-numeric string Ex. toupper

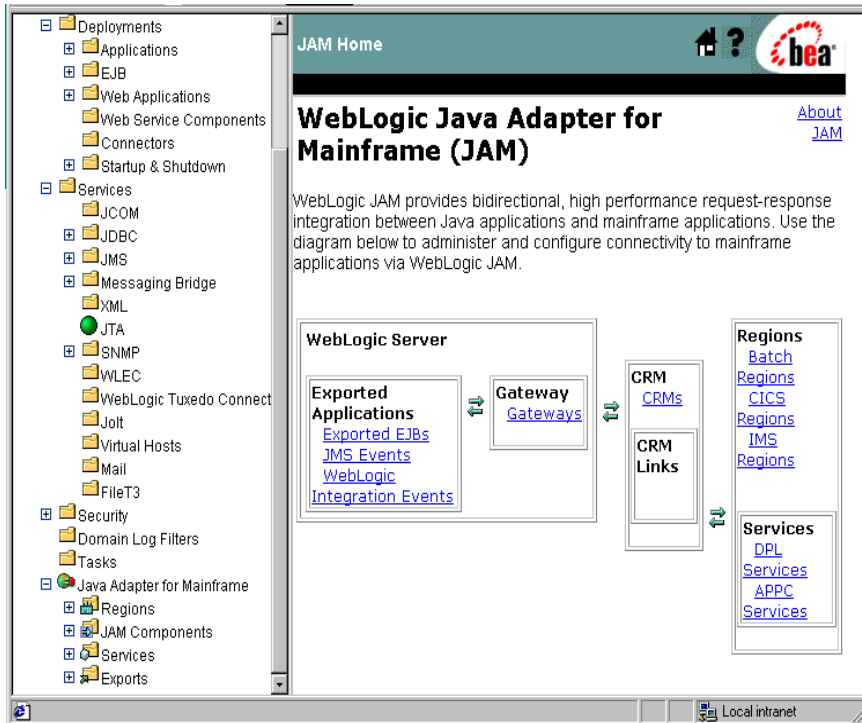
WebLogic Administration Console Parameter	Description	Parameter Syntax
<b>Transaction Program ID</b>	The transaction profile name associated with the transaction program.	Alpha-numeric string. First character must be an alpha character. Ex. TOUPPER
<b>Timeout:</b>	Approximate amount of time that the program usually takes to complete successfully. WebLogic JAM uses this timeout value to return a timeout error to the calling WebLogic Server application.	Numeric value in milliseconds Ex. 250
<b>Input and Output Schema (if using WebLogic Integration)</b>	<p>The schema names for this APPC program, if using WebLogic Integration.</p> <p><b>Note:</b> For information and instructions for creating an input schema with WebLogic Integration, see the <i>BEA WebLogic Java Adapter for Mainframe Workflow Processing Guide</i>.</p>	Alpha-numeric string Ex. emprec

#### Steps for Configuring APPC/MVS Transaction Programs as APPC Services

To create the configuration information that WebLogic JAM needs to invoke this APPC program, use the WebLogic Administration Console.

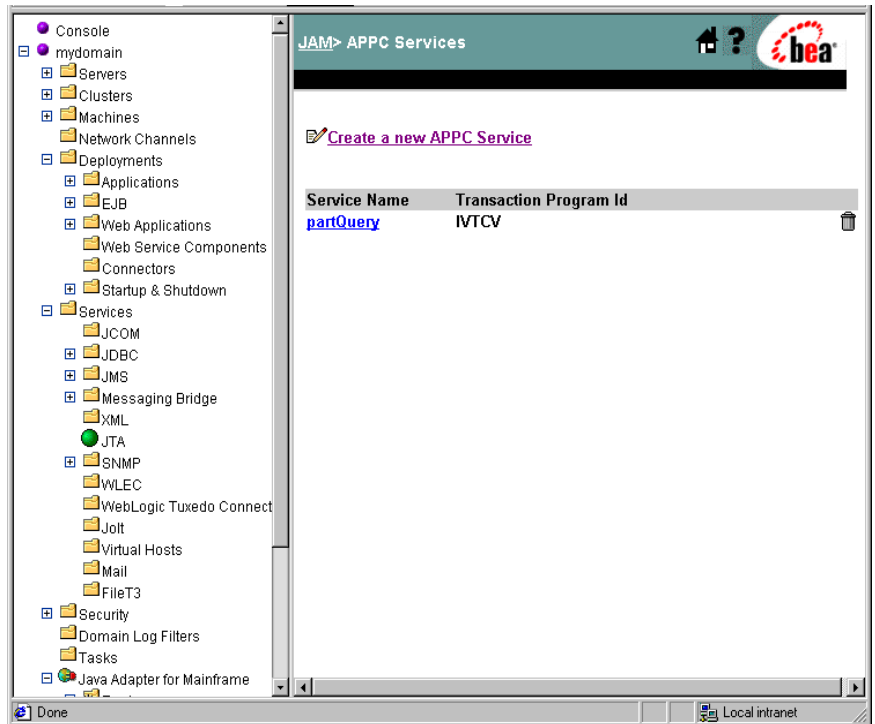


1. From the WebLogic JAM Home page, click **APPC Services**.

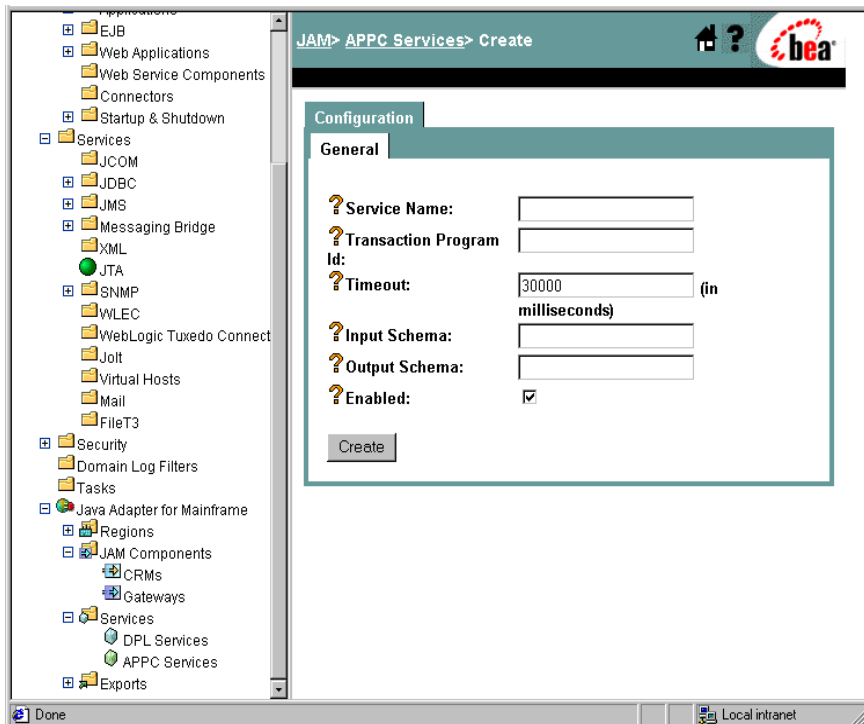


### 3 Integrating Applications With WebLogic JAM

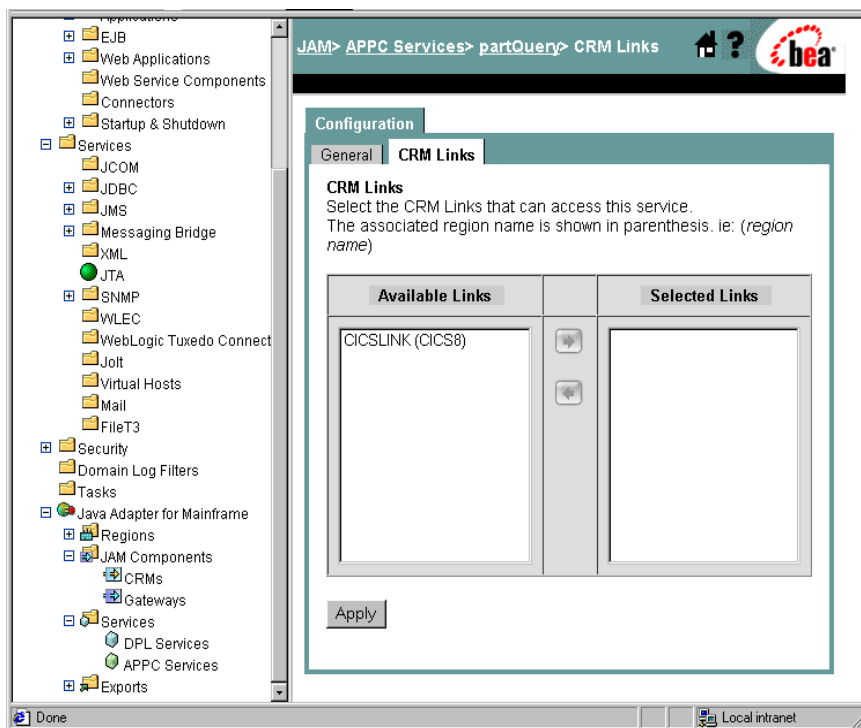
The List APPC Services page displays.



### 2. Click **Create a New APPC Service**.



3. Enter the configuration parameters in the appropriate fields. A check in the **Enabled** box makes the service available for use. Removing the check from the **Enabled** box disables the service. Click **Create** to submit your parameters.



4. Once you have created the APPC service definition in the administration console, you must select a link to the region where the APPC program is available. If the program is available in more than one mainframe region, you may select links to some or all of the regions in which it is present.

The left pane lists the regions that are defined in the WebLogic JAM configuration. The right pane lists the regions where this APPC program resides. Select the links from the list of all defined regions and move them to the list of regions where this APPC program is available.

5. Click Apply when you have finished adding regions.

# Exposing J2EE Applications to Mainframe Clients

WebLogic JAM enables mainframe applications to invoke J2EE applications by mapping a service name to a J2EE application. The WebLogic Administration Console provides you with a way to configure a mapping between a service name and a J2EE application (EJB, JMS, or WebLogic Integration application).

When WebLogic JAM receives a request from a mainframe application with the specified service name, WebLogic JAM invokes the corresponding J2EE application.

This section provides information on the following subjects:

- [Configuring Mainframe Client Applications](#)
- [Exposing Enterprise Java Beans to the Mainframe](#)
- [Exposing JMS Events to the Mainframe](#)

## Configuring Mainframe Client Applications

WebLogic JAM enables CICS DPL, standard IMS, and APPC programs (that run in CICS, IMS or as a batch job) as mainframe clients to invoke J2EE applications. Depending on the type of mainframe client application, some configuration may be required on the mainframe. This configuration usually involves creating a program or transaction definition that points to WebLogic JAM.

It will usually be necessary to convert the format of the data passed between WebLogic Server applications and mainframe applications. It may be necessary to collect information about data formats from a mainframe application (usually a copybook) for use in data conversion. For further information about using data conversion in your application, refer to the *BEA WebLogic Java Adapter for Mainframe Programming Guide*.

The following sections provide information on additional configuration for each different type of supported mainframe clients:

- [CICS Distributed Program Link Clients](#)

- [Standard IMS Programs](#)
- [APPC Programs that Run in CICS, IMS or as a Batch Job](#)

### **CICS Distributed Program Link Clients**

A CICS program can invoke a J2EE application by requesting a service configured for WebLogic JAM.

WebLogic JAM enables CICS programs on the mainframe to execute a `CICS EXEC LINK` call to a J2EE application as if it were a CICS program on a remote CICS region. The `PROGRAM` name parameter of the `EXEC CICS LINK` call may correspond to a CICS program resource definition or these may be coded directly on the `EXEC LINK` statement. These two approaches are described in the following sections:

- [Specifying the SYSID Parameter on the EXEC CICS LINK Call](#)
- [Using a CICS Program Resource Definition](#)

#### **Specifying the SYSID Parameter on the EXEC CICS LINK Call**

When the program link statement contains the `SYSID(systemname)` parameter, then the CICS program resource definition is not used. The values are taken directly from the `LINK` statement. [Listing 3-1](#) provides a sample CICS LINK statement with the `SYSID` parameter specified:

#### **Listing 3-1 Sample CICS LINK Statement**

---

```
EXEC CICS LINK
  PROGRAM (name)
  SYSID(systemname)
  COMMAREA(data-area)
  LENGTH(data-value)
```

---

The parameters used to send the request to WebLogic Server (J2EE) applications are:

**PROGRAM(name)**

The service name assigned to the WebLogic (J2EE) application, such as the exported EJB, JMS Event, or WebLogic Integration Event.

**SYSID(systemname)**

Defines the connection ID for the CRM logical unit that the service request is sent, and corresponds to the CICS connection resource definition ID.

### Using a CICS Program Resource Definition

An alternative to specifying the `SYSID` parameter in the `EXEC CICS LINK` call is to create a program resource definition in CICS. This program definition specifies `REMOTESYSTEM` and `REMOTENAME` parameters that point to WebLogic JAM. CICS applications can then specify the name of the program resource definition as the `PROGRAM` parameter of the `EXEC CICS LINK` call. Using this approach, CICS client programs do not have to be modified when the `SYSID` parameter changes; instead, only the program resource definition is changed.

[Listing 3-2](#) provides a sample CICS program definition with the `REMOTESYSTEM` and `REMOTENAME` parameters specified:

#### Listing 3-2 Sample CICS Program Definition

---

```
DEFINE PROGRAM (TRADE)
  GROUP(BEA)
  DESCRIPTION(DISTRIBUTED PROGRAM LINK DEFINITION)
  LANGUAGE(COBOL)
  REMOTESYSTEM(BEA1)
  REMOTENAME(TRADCLNT)
```

---

The parameters used to send the request to WebLogic Server (J2EE) applications are:

**PROGRAM(name)**

The name of the program definition. This name must match the name used on the `EXEC CICS LINK` definition.

**REMOTESYSTEM(name)**

The name of the remote system. This is the connection ID name that is defined for the CRM logical unit. For more information about the connection definition, refer to [“Configuring WebLogic JAM Connectivity.”](#)

**REMOTENAME(name)**

The service name assigned to the WebLogic (J2EE) application, such as the exported EJB, JMS Event, or WebLogic Integration Event.

## Standard IMS Programs

Standard IMS applications can use the DL/I application-programming interface to send output messages to J2EE applications. IMS provides implicit APPC support that translates DL/I calls into APPC communications. Using this implicit APPC support, standard IMS applications can insert output messages to J2EE applications.

IMS applications must issue a `Change (CHNG)` call to change its output destination to an alternate Program Control Block (PCB). This alternate PCB can specify an LU 6.2 descriptor that points to WebLogic JAM. Then an output message can be inserted by issuing an `Insert (ISRT)` call. This call forwards the output message to WebLogic JAM, which then sends the message to the configured J2EE application.

**Note:** WebLogic JAM does not support the insertion of multiple output messages; it only supports a single output message per invocation.

Work closely with your IMS administrator to define an LU 6.2 descriptor with the following parameters:

**LTERM**

The **LTERM** is an arbitrary name. The IMS program specifies this name as the destination in the alternate Program Control Block (PCB).

**LUNAME**

The logical unit of the CRM.

**TPNAME**

The service name configured in WebLogic JAM to point to the desired J2EE application.

**SYNCLEVEL**

Specify this parameter as **N**. Two-phase commit is not supported for IMS client programs that use implicit APPC.

**Note:** For more information on configuring LU 6.2 descriptors, see IMS documentation.

## APPC Programs that Run in CICS, IMS or as a Batch Job

An APPC program can invoke a J2EE application by using CPI Communications (CPI-C), APPC verbs, or CICS DTP commands. WebLogic JAM supports a single request/response conversation between the APPC program and the specified J2EE



application. For more information on how WebLogic JAM supports single request/response conversations with APPC programs, see ["Programming Flows"](#) in the *BEA WebLogic Java Adapter for Mainframe Programming Guide*.

Some configuration is recommended when you use APPC to connect to a J2EE application. An APPC program that connects to a J2EE application must identify that it wants to connect to a remote system and a remote partner transaction. Rather than hard coding these parameters in the APPC program, the recommended approach is to store these parameters in side information.

Side information allows an APPC program to identify its partner program with a symbolic destination name. This symbolic destination name is specified in the APPC program, and the remote system and remote partner transaction are configured in the correspondingly named side information. This allows the remote system and remote partner transactions to change without having to modify the APPC program.

Use the following information to help your system administrator configure side information to connect to J2EE applications using WebLogic JAM.

### CICS Side Information

CICS provides a Partner Resource Definition for defining side information. The Partner Resource Definition should define the following:

**NETNAME**

The logical unit of the CRM.

**PROFILE**

The name of the communication profile that specifies additional parameters of the communications between the logical units involved. Using the CICS supplied profile DFHCICSA is recommended.

**TPNAME**

The name of the service configured in JAM that points to the desired J2EE application.

For more information on creating CICS partner resource definitions see CICS documentation.

### APPC/MVS Transaction Program Profile Definition

IMS and APPC/MVS batch implementations of APPC provide the following side information parameters.

**Partner LU Name**

The logical unit of the CRM.

**Transaction Program Name**

Specifies the name of the remote program. This should be set to the service name configured in WebLogic JAM that points to the desired J2EE application.

**Mode Name**

This should be set to the mode name used in the CRM link that connects to this application.

## Exposing Enterprise Java Beans to the Mainframe

Mainframe applications can invoke eGen-generated session Enterprise Java Beans (EJBs). When WebLogic JAM receives a request from a mainframe application for this EJB, WebLogic JAM looks up the EJB and invokes it on behalf of the mainframe client.

See the *BEA Java Adapter for Mainframe Programming Guide* for more information on creating an EJB using the eGen utility.

To make an EJB available to mainframe applications, you need the following information:

- [Steps for Creating an Export Definition for an EJB](#)

## WebLogic Administration Console Parameters for Enterprise Java Beans Exposed to the Mainframe

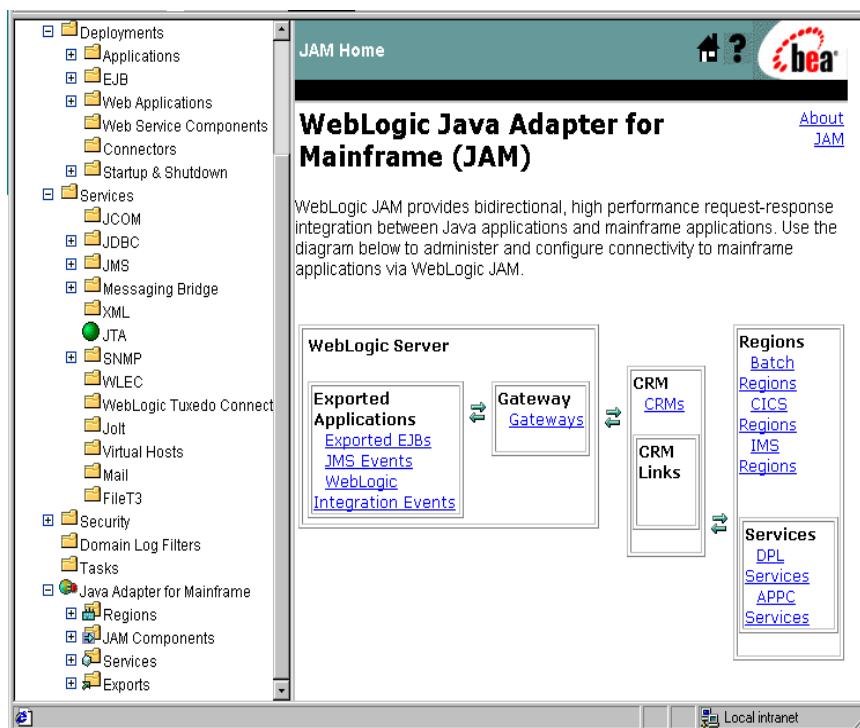
WebLogic JAM requires the parameters shown in [Table 3-4](#) to expose a EJBs to the mainframe.

**Table 3-4 EJB Parameters**

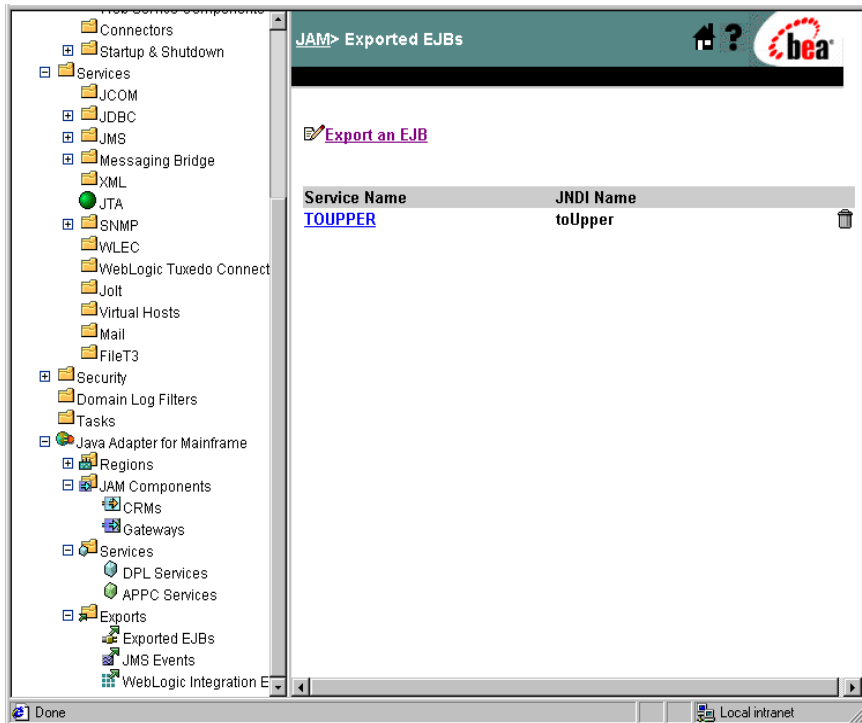
<b>WebLogic Administration Console Parameter</b>	<b>Description</b>	<b>Parameter Syntax</b>
<b>Service Name</b>	The name that is specified by mainframe applications when they want to invoke this EJB.	Alpha-numeric string Ex. toupper
<b>JNDI Name</b>	The JNDI name used to look up the home interface of this EJB.	Alpha string Ex. application.good.toupper
<b>Timeout:</b>	Approximate amount of time that this EJB usually takes to complete successfully. WebLogic JAM uses this timeout value to return a timeout error to the calling mainframe application.	Numeric value in milliseconds Ex. 250

## Steps for Creating an Export Definition for an EJB

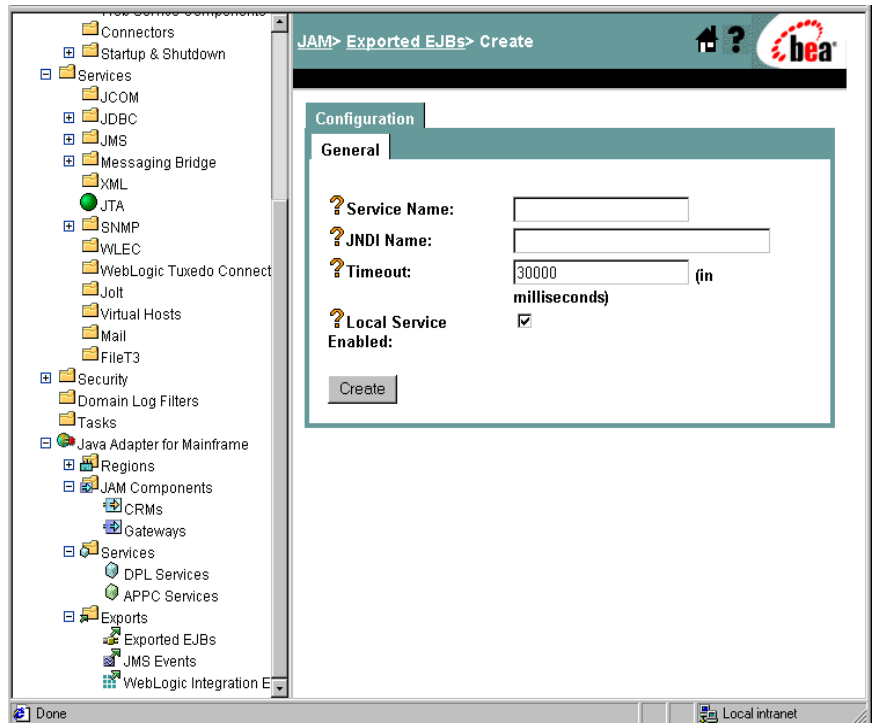
1. From the WebLogic JAM Home page, click **Exported EJBs**.



The Exported EJBs page displays.



2. Click **Export an EJB**.



3. Enter the configuration parameters in the appropriate fields. A check in the **Local Service Enabled** box makes the service available for use. Removing the check from the **Local Service Enabled** box disables the service. Click **Create** to submit your parameters.
4. To configure a mainframe application to invoke this service, see the instructions in [“Configuring Mainframe Client Applications.”](#)

## Exposing JMS Events to the Mainframe

WebLogic JAM can enqueue data sent by mainframe applications to a JMS queue or topic. WebLogic JAM can use a Java class called a `DataView` to translate the mainframe data to XML. WebLogic JAM then enqueues the data to the specified JMS queue or topic.

See the *BEA Java Adapter for Mainframe Programming Guide* for more information on generating a `DataView` class.

Use the WebLogic Administration Console to perform the following configuration tasks for creating JMS definitions:

- [Steps for Creating a JMS Event Definition](#)

### WebLogic Administration Console Parameters for JMS Events Exposed to the Mainframe

WebLogic JAM requires the parameters shown in [Table 3-5](#) to expose a JMS events to the mainframe.

**Table 3-5 JMS Parameters**

WebLogic Administration Console Parameter	Description	Parameter Syntax
<b>Service Name</b>	The name that is specified by mainframe applications when they want to enqueue data to this JMS destination.	Alpha-numeric string Ex. <code>toupper</code>
<b>JMS Destination Type</b>	The type of JMS destination (either JMS Queue or JMS Topic).	Select from provided list. Ex. <code>application.toupper.queue</code>

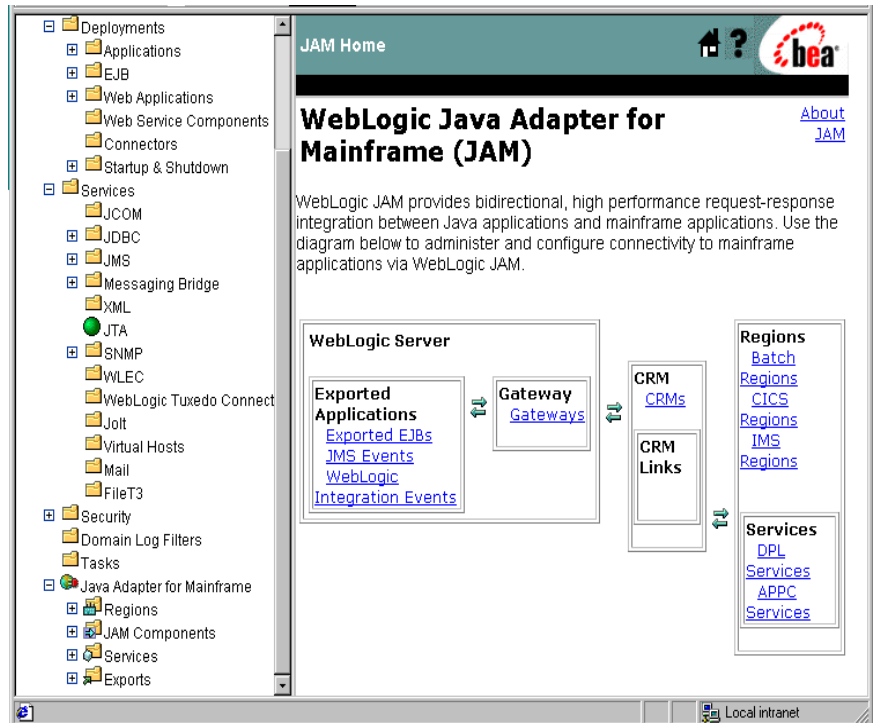
WebLogic Administration Console Parameter	Description	Parameter Syntax
<b>JMS Destination:</b>	The JNDI name of the JMS destination where the data sent by the mainframe application is to be enqueued.	Alpha string Ex. application. good.toupper
<b>Predefined JMS Destination</b>	This drop-down menu lists all JMS destinations currently defined within your WebLogic configuration. Selecting one of these destinations automatically populates the JMS Destination Type and JMS Destination parameters.	Select from the provided list.
<b>JMS Connection Factory</b>	The JNDI name of the JMS connection factory that is to be used to create a connection to the JMS destination. Different JMS connection factories produce JMS connections that support different features.  For example an XA JMS connection factory supplies JMS connections that are transactional and support XA transaction semantics.	Alpha string Ex. xA
<b>Predefined JMS Connection Factories</b>	The drop-down menu lists all of the JMS connection factories currently defined within your WebLogic configuration. Selecting one of these factories automatically populates the JMS connection factory parameter.	Select from the provided list.
<b>Dataview</b>	The fully qualified class name of a DataView class that is used to transform the binary data received from the mainframe application into XML. Refer to the <i>BEA WebLogic Java Adapter for Mainframe Programming Guide</i> for more information.  This DataView class must be in the WebLogic CLASSPATH. If not specified, the binary data that the mainframe sends is enqueued without translation.	Alpha string Ex. MyView



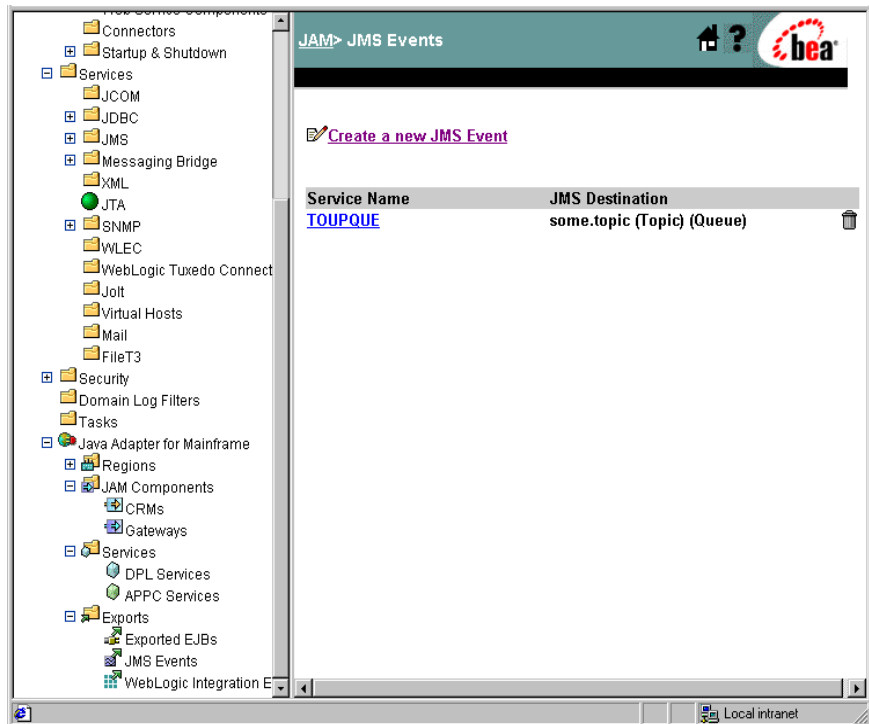
<b>WebLogic Administration Console Parameter</b>	<b>Description</b>	<b>Parameter Syntax</b>
<b>Timeout</b>	Approximate amount of time that the data transformation and enqueueing process should take to complete successfully.	Numeric value in milliseconds Ex. 250
<b>JMS Properties</b>	An optional parameter that lets you specify string properties that are to be set as additional headers in the JMS message that WebLogic JAM creates. You can set these properties as application-specific headers in the JMS message. You can also specify the JMS-defined JMSType.	Alpha string

## Steps for Creating a JMS Event Definition

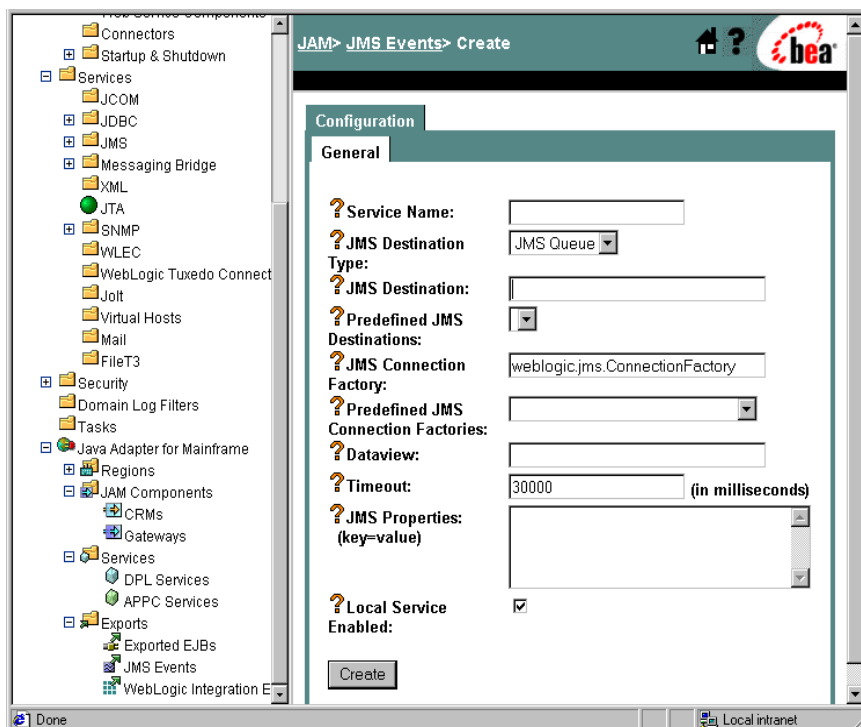
1. From the WebLogic JAM Home page, click **JMS Events**.



The JMS Events page displays.



2. Click **Create a New JMS Event**.



3. Enter the configuration parameters in the appropriate fields. A check in the **Local Service Enabled** box makes the service available for use. Removing the check from the **Local Service Enabled** box disables the service. Click **Create** to submit your parameters.
4. To configure a mainframe application to invoke this service, see the instructions in [“Configuring Mainframe Client Applications.”](#)

# 4 Scaling WebLogic JAM Configurations

WebLogic JAM can be configured to scale as your integration needs grow. WebLogic JAM can also be configured with redundant components that can fail over when a hardware or software system fails. Reliability and scalability are goals of any mission critical application.

- **Reliability**

the ability of the system to continue functioning under adverse conditions such as server or network outage

- **Scalability**

the ability of the system to provide acceptable throughput and performance under high volume use.

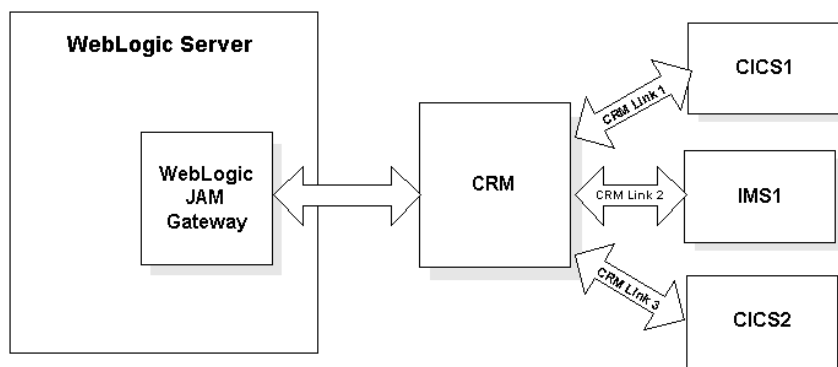
In support of these mission critical application goals, WebLogic JAM offers the following features:

- [Connecting to Multiple Regions](#)
- [Connecting Multiple Gateways to a Single CRM](#)
- [Support for a Clustered Environment](#)
- [Support for Distributed Transactions](#)

## Connecting to Multiple Regions

The CRM component of WebLogic JAM uses APPC to establish connections to back-end systems. A single CRM can establish multiple CRM Links (APPC connections) to different back-end systems. When a mainframe application is available in multiple back-end systems, the CRM can load-balance service requests between the back-end systems and can failover if a connection to a back-end system is lost.

**Figure 4-1 CRM Connection to Multiple Regions**



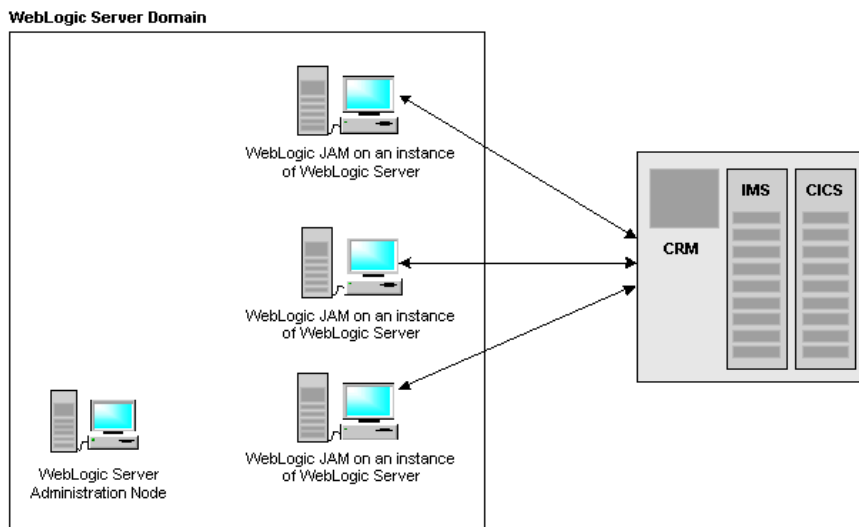
## Connecting Multiple Gateways to a Single CRM

WebLogic JAM CRM permits multiple Gateways to connect and use the resources provided by a single CRM process. This feature considerably reduces the processing load required on the computer hosting the CRM while supporting distribution of workload at the WebLogic JAM Gateway level. Configuration of a single CRM is easier than the configuration of multiple CRMs.

Multi-Gateway configuration is common for installations where a number of WebLogic JAM Gateways are hosting resources for a single mainframe. The multiple Gateways provide mainframe to WebLogic load balancing and failover. [Figure 4-2](#) shows multi-Gateway configuration in the WebLogic Server domain.

**Note:** For more information about support for load balancing and failover in a clustered environment, see [“Support for a Clustered Environment.”](#)

**Figure 4-2 Multi-Gateway/Single CRM Scenario**



The CRM provides support for multiple, concurrently connected instances of WebLogic Server / WebLogic JAM Gateway.

This section provides information about the following topics:

- [Advantages of Multi-Gateway Support](#)
- [Disadvantages of Multi-Gateway Configuration](#)
- [Multi-Gateway Connection Issues](#)
- [Multi-Gateway Disconnect and Shutdown Issues](#)
- [Understanding Mainframe to WebLogic Load Balancing in a Multi-Gateway CRM](#)

# Advantages of Multi-Gateway Support

The primary advantage of Multi-Gateway support is the ability for the CRM to host concurrent connections from one or more WebLogic JAM Gateways. The advantages of this approach include:

- This configuration supports clustering of WebLogic Servers and their associated WebLogic JAM Gateways.
- Overall, a single CRM consumes less system resources than multiple CRMs.
- The CRM has a single configuration which is common to all WebLogic Servers.
- The resulting configuration is much easier for you to configure, administer, and operate.
- The CRM can load balance mainframe requests for services provided by the connected WebLogic Servers.

For more information about WebLogic JAM support for clustered environment, load balancing, transaction management, and bi-directional failover support, see [“Support for a Clustered Environment.”](#)

# Disadvantages of Multi-Gateway Configuration

In a multiple Gateways CRM configuration, the CRM is a single point of failure. To overcome this disadvantage, multiple CRMs and multiple Gateways can be configured to provide more redundancy and reliability to your WebLogic JAM system. This is described further in [“Clustered Gateways, Multiple CRMs.”](#)

# Multi-Gateway Connection Issues

The CRM offers a single point of contact – that is, a single listener socket (identified by host IP address and port number) to which components (such as WebLogic JAM Gateways) can address their connection requests. Each new connection request is handed off to a server socket dedicated to servicing that component connection.



Version checking ensures that the CRM supports and is compatible with the WebLogic JAM component. If the component does not meet these requirements, the connection request is rejected. All concurrently connected WebLogic JAM Gateways have equal standing with respect to the CRM, thus making them equal peers.

However, the CRM attaches significance to the following component connect/disconnect scenarios:

- [First WebLogic JAM Gateway Connection](#)
- [Subsequent WebLogic JAM Gateway Connections](#)

### First WebLogic JAM Gateway Connection

If the component is a WebLogic JAM Gateway and this is the first (or only) Gateway to connect to the CRM, the CRM accepts configuration information from the WebLogic Administration Console to initialize the VTAM interface and to establish links with mainframe systems, such as CICS and IMS.

When CRM configuration is complete, the client receives a response indicating that the CRM is ready to begin processing requests. This configuration is the common deployment configuration for WebLogic JAM.

For detailed information about configuring connectivity for the CRM, see [“Step 2: Define Where the CRM Will Run”](#) in [“Configuring WebLogic JAM Connectivity.”](#)

### Subsequent WebLogic JAM Gateway Connections

If the client is a WebLogic JAM Gateway and there are other Gateways currently connected to the CRM, the client is held in abeyance (if necessary) until CRM configuration is complete. In this case, the first WebLogic JAM Gateway client has already supplied the CRM configuration information and configuration information is not required. If and when CRM configuration is complete, the client receives a response indicating that the CRM is ready to begin processing requests.

## Multi-Gateway Disconnect and Shutdown Issues

WebLogic JAM provides support for the following orderly disconnection and shutdown procedures.

- [Gateway Disconnect](#)
- [Shutdown Processing \(All Gateways Except the Last\)](#)
- [Shutdown Processing \(Last Gateway\)](#)

### **Gateway Disconnect**

When a WebLogic JAM Gateway disconnects, the CRM continues to service other connected Gateways. When the last (or only) WebLogic JAM Gateway disconnects, the CRM resets its current configuration, disconnecting from any mainframe systems (CICS, IMS) with which it has established connections.

### **Shutdown Processing (All Gateways Except the Last)**

During shutdown, a Gateway disconnects (closes the socket connection), Phase 1 shutdown processing is initiated for that connection only. Phase 1 shutdown is a “shutdown pending” state in which the following is true:

- No new mainframe to WebLogic requests are routed to that client.
- Sends and receives in progress are permitted to complete.
- In-flight transactions are aborted.

The CRM will not proceed to Phase 2 shutdown processing if other Gateways are still connected. Links to mainframe systems will remain active and the CRM will continue normal processing for other connected Gateways.

### **Shutdown Processing (Last Gateway)**

When the last (or only) Gateway requests shutdown or closes its connection with the CRM, the CRM executes Phase 1 shutdown processing for that Gateway as described in [“Shutdown Processing \(All Gateways Except the Last\).”](#) However, when Phase 1 shutdown processing is complete, the CRM proceeds to Phase 2 shutdown processing, in which the following occurs:

- Links to mainframe systems are stopped.
- The current configuration is discarded.
- The CRM returns to a “reset” state pending connection by another client.

**Note:** If and when a connection request is subsequently received from a WebLogic JAM Gateway, the CRM will re-configure itself. For detailed information about CRM startup and shutdown, see [“CRM Administration.”](#)

## Understanding Mainframe to WebLogic Load Balancing in a Multi-Gateway CRM

Multi-Gateway support allows several WebLogic JAM Gateways to be concurrently connected to a single CRM. The connected Gateways share a common configuration and hence offer a common set of services. Therefore, a mechanism must exist to distribute mainframe to WebLogic requests (originating on the mainframe) among the available gateways, a technique commonly referred to as load balancing. Normally, the objective of load balancing is to distribute requests for service across the available servers in such a way that serialization is minimized, resource requirements are balanced, and overall throughput is maximized.

Many algorithms exist for scheduling work among multiple processors, ranging from the very simplistic to the very complex. Mainframe to WebLogic requests are always load balanced via the round-robin method.

Mainframe to WebLogic request load balancing is a function of the WebLogic JAM CRM process. The CRM load balances between all connected gateways without regard to their cluster arrangement.

## Support for a Clustered Environment

A WebLogic cluster is a group of servers that, from the application point-of-view, operate as a single server. A cluster provides:

- **Scalability**

additional servers can be added to the cluster dynamically to increase capacity.

- **High-availability**

redundancy of multiple servers insulates applications from failures

The cluster support for the WebLogic JAM Gateway enables reliable remote access to mainframe services. Clients may run on any machine with network access to the WebLogic Server machine(s) hosting the Gateway and benefit from transparent load balancing and failover features offered by both WebLogic Server and WebLogic JAM.

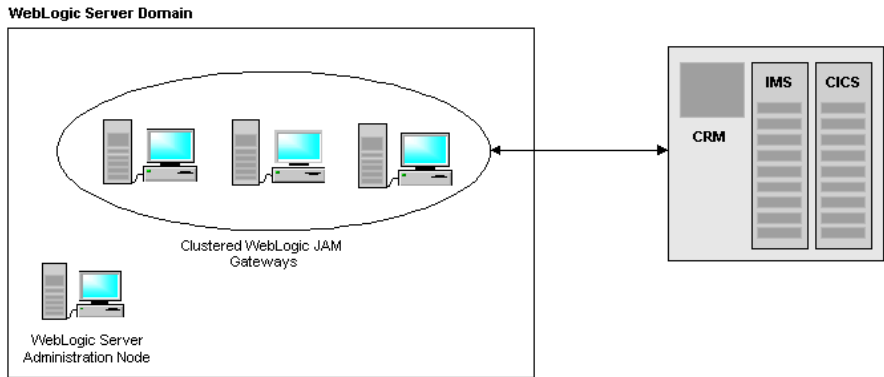
WebLogic JAM Gateway offers client access to the mainframe service via the Java Remote Method Invocation (RMI) feature of WebLogic Server. RMI access allows WebLogic JAM to take advantage of all the cluster features of WebLogic Server while still maintaining a level of control for enabling/disabling services.

WebLogic JAM provides the following features in a clustered environment:

- [Clustered Gateways, Single CRM](#)
- [Clustered Gateways, Multiple CRMs](#)
- [Load Balancing Support with WebLogic Server](#)
- [Failover Support with WebLogic Server](#)

### **Clustered Gateways, Single CRM**

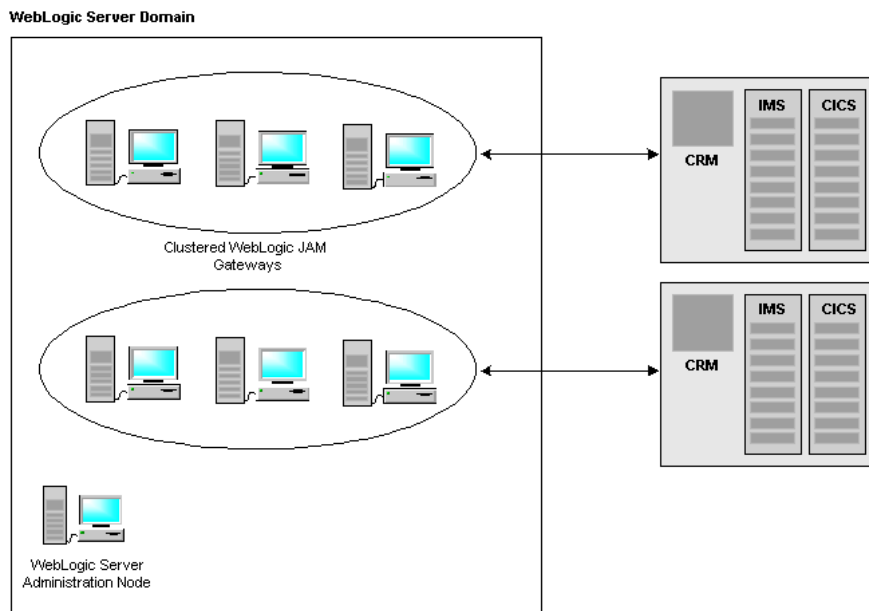
A scenario of clustered Gateways with a single CRM is recommended for large sites where a number of WebLogic JAM Gateways are deployed to access the resource in a single mainframe. Load balancing and failover are supported for both mainframe to WebLogic and WebLogic to mainframe requests while minimizing mainframe resources used. [Figure 4-3](#) shows clustered multi-Gateway configuration in a WebLogic Server domain.

**Figure 4-3 Clustered Gateways/Single CRM Scenario**

## Clustered Gateways, Multiple CRMs

In this environment, multiple WebLogic JAM Gateways are deployed in a cluster, and the Gateways are distributed to two different CRMs. This configuration offers the flexibility to deploy Gateways as required. In this clustered environment, each CRM configuration should be homogeneous, offering the same back-end application services. Multiple CRMs offer additional failover capability if they are connected to the same back-end systems. In this configuration, half of the Gateways are assigned to each CRM to provide maximum redundancy. However, a combination of multiple Gateways to a CRM and single Gateway to a CRM can be combined. [Figure 4-4](#) shows a clustered multi-Gateway configuration in a WebLogic Server domain.

**Figure 4-4 Clustered Gateways/Multiple CRM Scenario**



## Load Balancing Support with WebLogic Server

The WebLogic Administration Console provides you with the tools to configure WebLogic to mainframe load balancing for WebLogic JAM. Load balancing is one of the primary tools employed in making systems scalable. WebLogic JAM takes advantage of the sophisticated load balancing features of WebLogic clusters for WebLogic to mainframe requests such that all WebLogic to mainframe load-balancing in WebLogic JAM occurs among a group of servers arranged in a WebLogic cluster. When requests are made to remote WebLogic JAM services, load balancing occurs among all WebLogic JAM Gateways in the cluster that offer the service.

For additional information on WebLogic Clusters please refer to the WebLogic Server documentation at <http://edocs/wls/docs70/cluster/index.html>.

## **Failover Support with WebLogic Server**

Failover is the ability of a system to detect a service failure and automatically retry the operation using another Gateway. This includes the ability to recover in the event of a server failure and the ability to find another instance of a service on an available server.

Under the best of conditions the results of a service are idempotent, meaning that multiple invocations of the service, with the same input, produce the same output with no side effects. An example of an idempotent service would be a currency translation service. The service produces the same output each time it is run for a given input value and it has no side effects. Under these conditions it is safe to failover a request for any type of failure.

### **Failover for WebLogic to Mainframe Services**

All WebLogic to mainframe services in WebLogic JAM are assumed to be non-idempotent. Non-idempotent services are programs that have effect on external events or that may return a different answer each time they are invoked. An example of a non-idempotent service is a program which produces a check to be mailed to a supplier. If the service fails after producing the check a retry operation would produce a second check. For non-idempotent services failover is only safe if it can be determined that the failure occurred before the service began processing the request.

### **Failover for Mainframe to WebLogic Services**

A mainframe → WebLogic service request consists of two parts. First a Query Service request is forwarded to a WebLogic JAM Gateway requesting the availability of the requested service. After receiving an affirmative response to the Query Service a Service Invocation request is forwarded to the Gateway. If a negative response is returned from the Query Service failover occurs to the next available Gateway. Failures returned from a Service Invocation are not failed over but are returned to the requesting application.

For detailed information about WebLogic Server failover support, refer to the “Failover Support for Clustered Services” section in the WebLogic Server documentation at <http://e-docs.bea.com/wls/docs70/cluster/index.html>.

# Support for Distributed Transactions

WebLogic JAM leverages distributed transaction support from WebLogic Server. Transactions allow multiple updates to one or more resources performed with integrity. For example, transferring from one bank account to another.

**Note:** For information about administering transaction, see “[Administering Transactions](#).”

WebLogic JAM offers full support for XA distributed transactions allowing resources on WebLogic Server, CICS, and IMS to be enlisted in a single tightly coupled transaction or multiple loosely coupled transactions. Regardless of the origin of a transaction the XA two-phase commit protocol supported by WebLogic JAM ensures the integrity of all resources effected.

In the case of loosely coupled transactions, the local transaction manager treats each transaction as distinct, preventing access to modifications. Although these transactions are loosely coupled, the transaction manager is not aware of the relationship between the multiple transactions.

Transactions, like requests and responses, are implicitly associated with a single CRM client. With multi-Gateway CRM support, all Gateways and regions connected to the CRM participate in the transaction management to ensure a tightly-coupled transaction in the global transaction.

For detailed information about WebLogic Server support for distributed transactions, refer to “Introducing Transactions” in the WebLogic Server documentation at <http://e-docs.bea.com/wls/docs70/jta/gstrx.html>.

## Transactional Affinity

When a transaction is started in a specific transaction manager or resource manager, then the transaction is referred to as having affinity to those resources. These resources are chosen over all other resources for subsequent requests in order to allow the transaction resources to be *tightly-coupled*. In the WebLogic JAM distributed transaction, transaction affinity is maintained. This means that within a cluster,



WebLogic will try to select the same server instance (and thus the same WebLogic JAM instance) to process all client requests in a transaction. In addition, a CRM will attempt to select the same CRM link to process all client requests in a transaction.

In order to ensure that the same server instance and/or CRM link are selected, then all services that participate in the transaction should be deployed in the same server instance or on the same CRM link. If this is not true, then a different instance or link is selected, and the transaction is said to be *loosely-coupled* with its resources.



# 5 Modifying Your Configuration

This section contains information on the following subjects

- [Modifying Your Connectivity Configuration](#)
- [Modifying Your Application Integration Configuration](#)

## Modifying Your Connectivity Configuration

Once you have configured WebLogic JAM connectivity, you have several options for modifying and updating connectivity information:

- [Modifying Region Definitions](#)
- [Modifying CRM Definitions](#)
- [Modifying CRM Link Definitions](#)
- [Modifying Gateway Definitions](#)

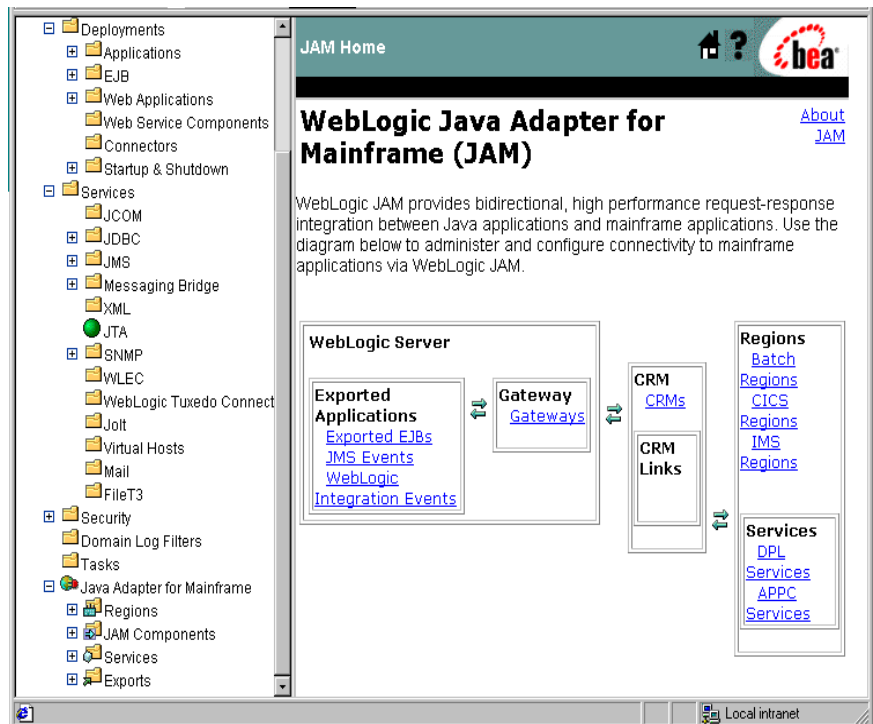
# Modifying Region Definitions

You can view and modify region definitions in the following ways:

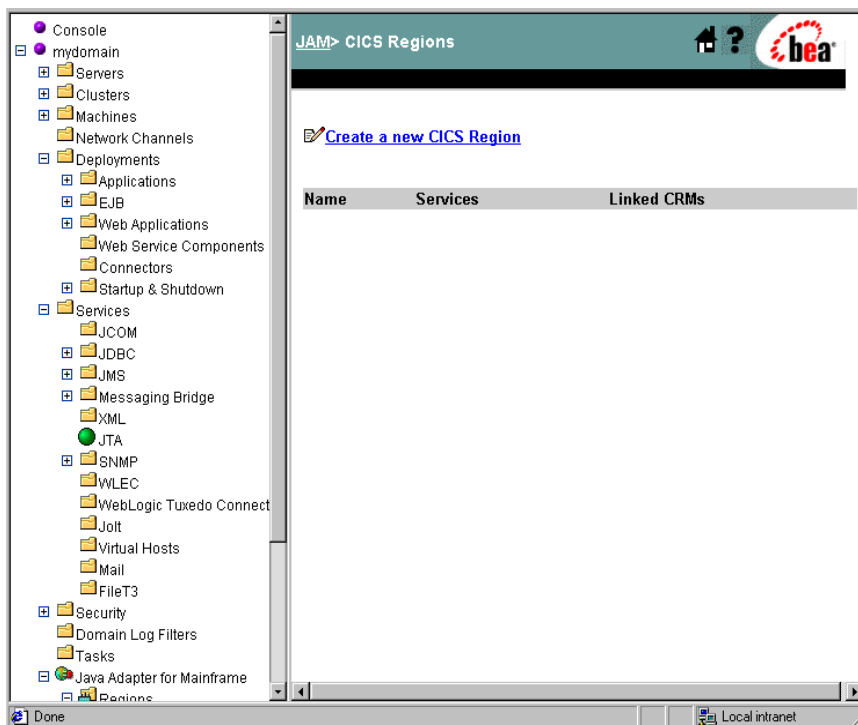
- [Listing Region Definitions](#)
- [Modifying the Logical Unit Name in a Region](#)
- [Deleting a Region Definition](#)

## Listing Region Definitions

To list existing region definitions, perform the following steps:

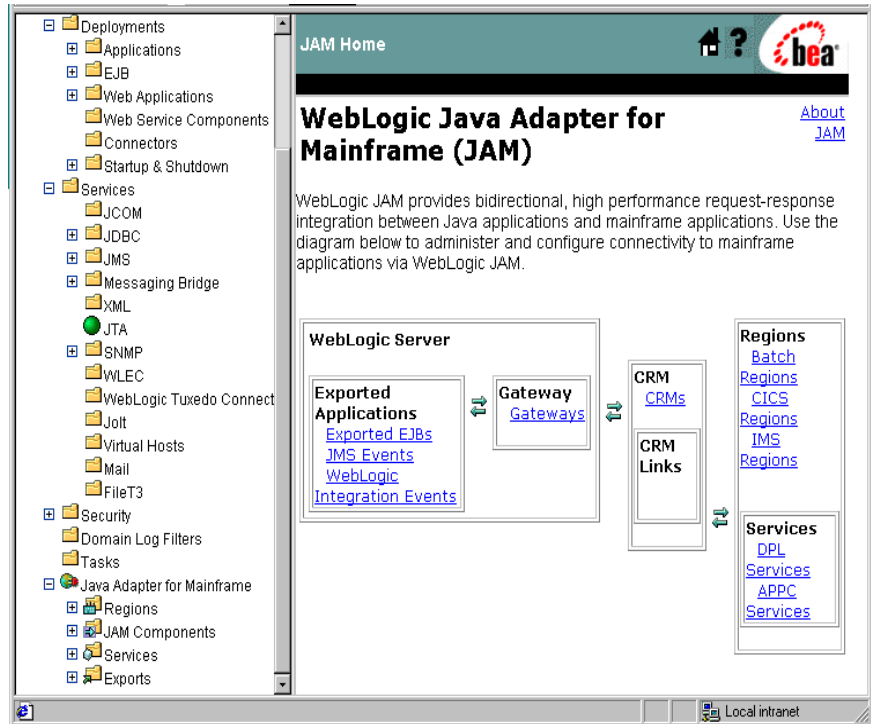


1. From the WebLogic JAM home page, click the region type for which you want to edit a definition (Batch, CICS, IMS). The List Regions Page displays.

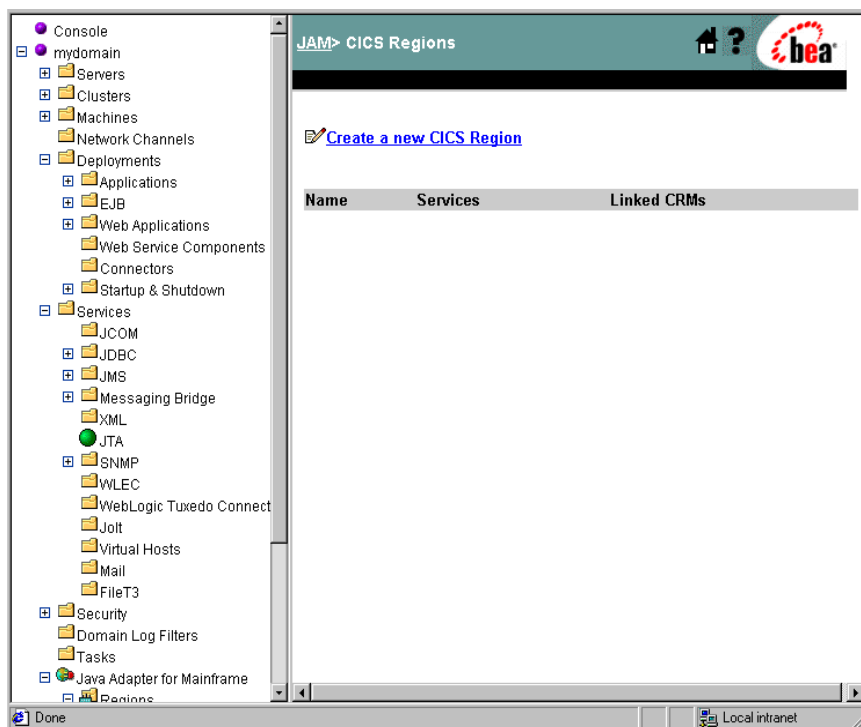


### Modifying the Logical Unit Name in a Region

To edit a region definition, perform the following steps:

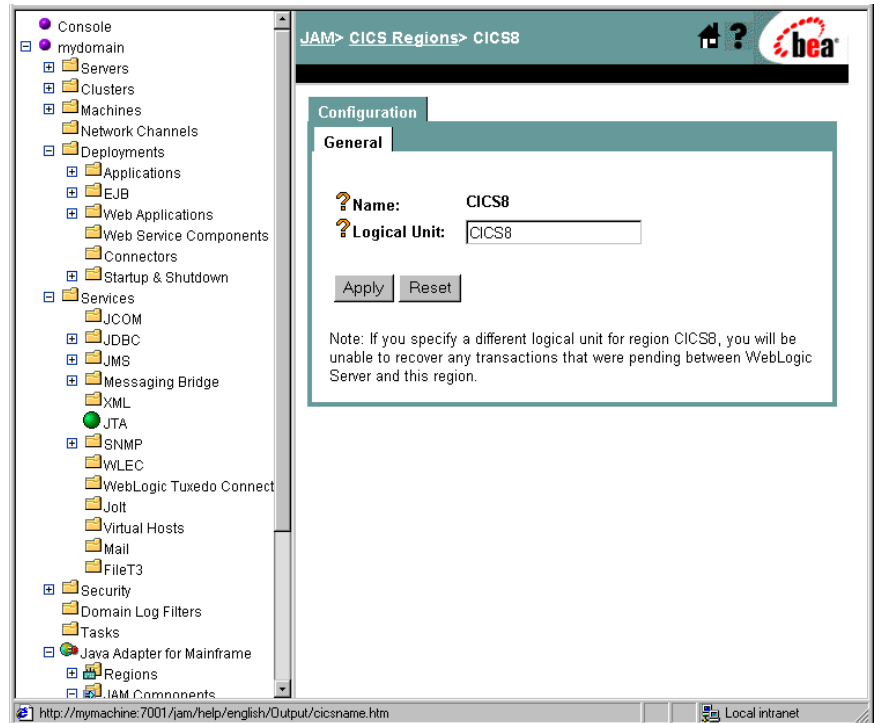


1. From the WebLogic JAM home page, click the region type for which you want to edit a definition (Batch, CICS, IMS). The List Regions Page displays.



2. Click the region definition that you want to edit.

## 5 Modifying Your Configuration



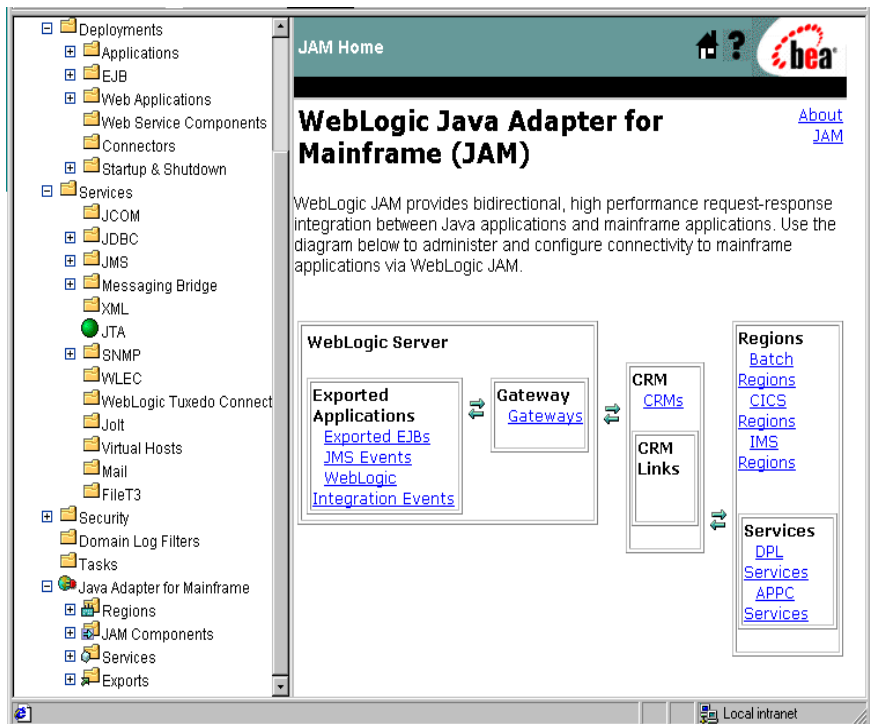
3. Enter the new Logical Unit parameter.
4. Click **Apply** to save your changes.
5. Cold start the link. For information and instructions on how to cold start the CRM link, see [“Cold Start.”](#)
6. Restart all WebLogic JAM components that access the region.
  - For information on restarting the CRM, see [“Starting the CRM.”](#)
  - For information on restarting a Gateway, see [“Starting and Stopping a Gateway.”](#)



### Deleting a Region Definition

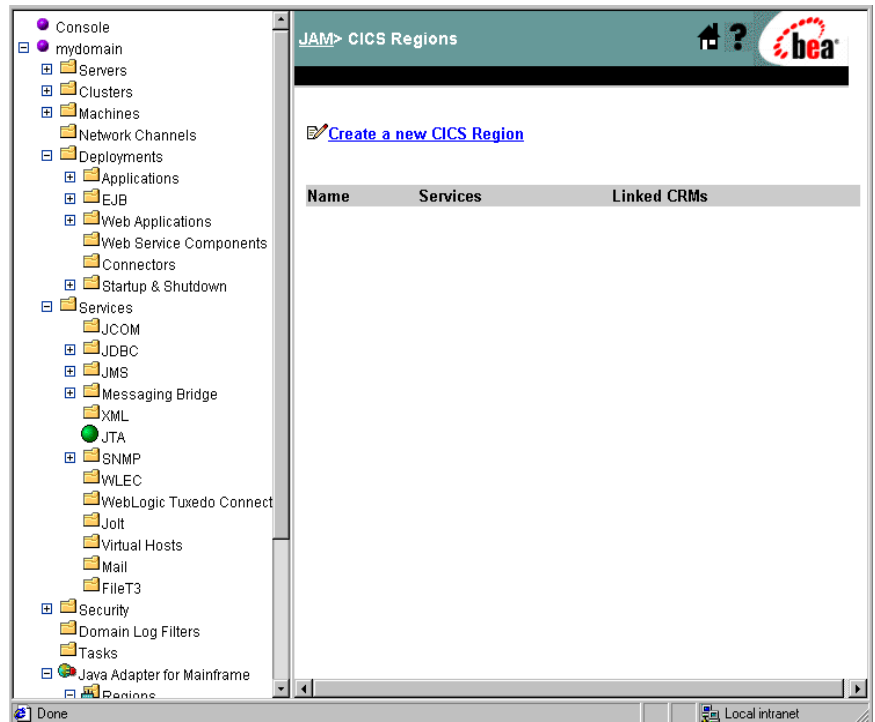
You can delete a region definition by selecting the associated trash can icon in the WebLogic Administration Console. The trash can icon is enabled only when the region has no linked CRMs. If you try to delete a region that is referenced by a CRM Link entity in the WebLogic JAM configuration, an error message displays, and the region is not deleted.

To delete a region definition, perform the following steps:

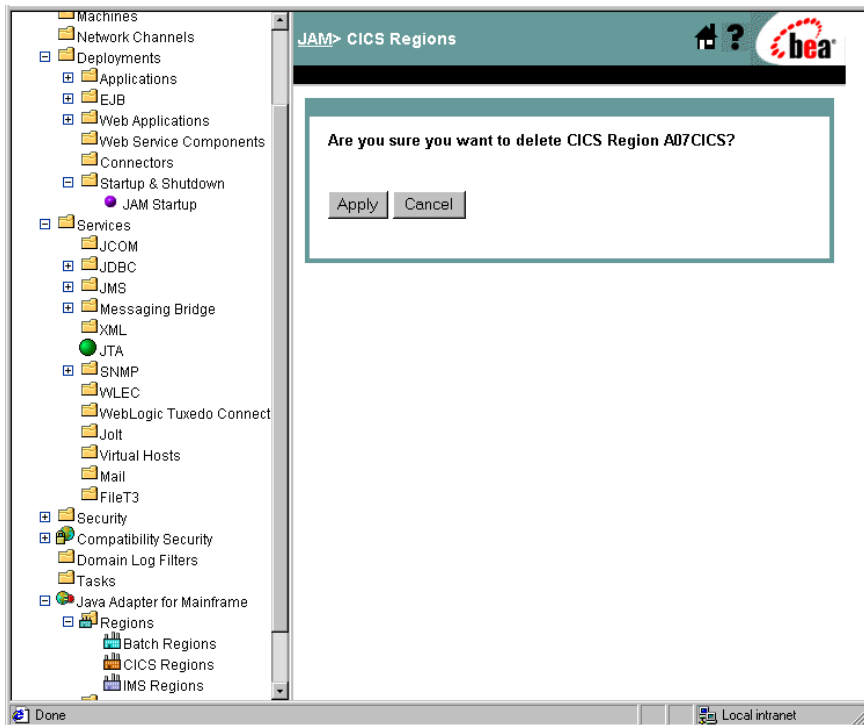


## 5 Modifying Your Configuration

1. From the WebLogic JAM home page, click the region type from which you want to delete a definition (Batch, CICS, IMS). The List Regions Page displays.



2. Delete any CRM Links that connect to the region to be deleted. For information about deleting CRM Links, see “[Deleting a CRM Link.](#)”
3. Click the trash can icon.



4. Click **Apply** to finish deleting the region definition.

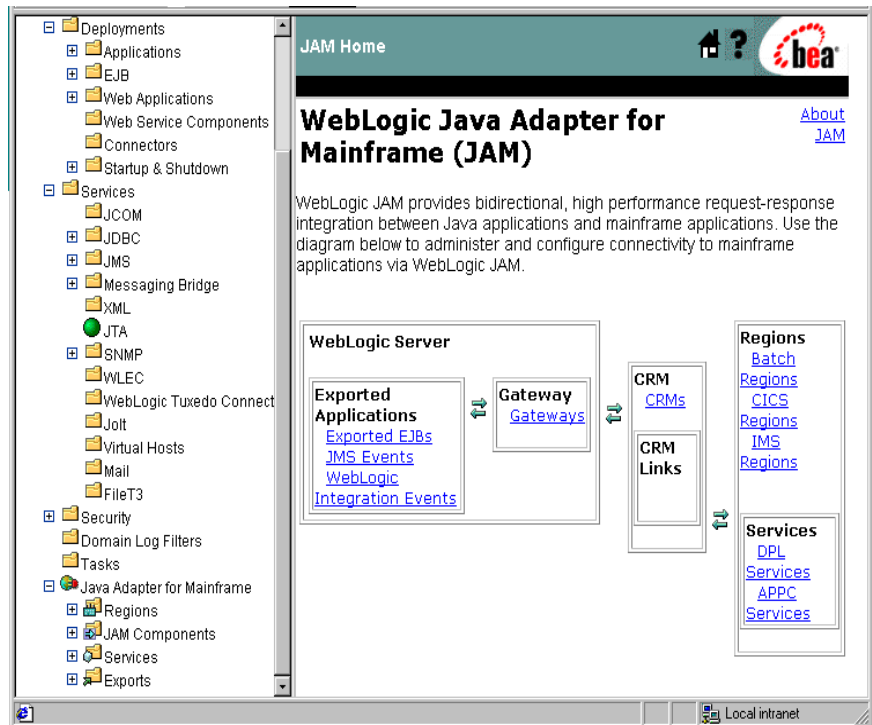
## Modifying CRM Definitions

You can view and modify existing CRM definitions in the following ways:

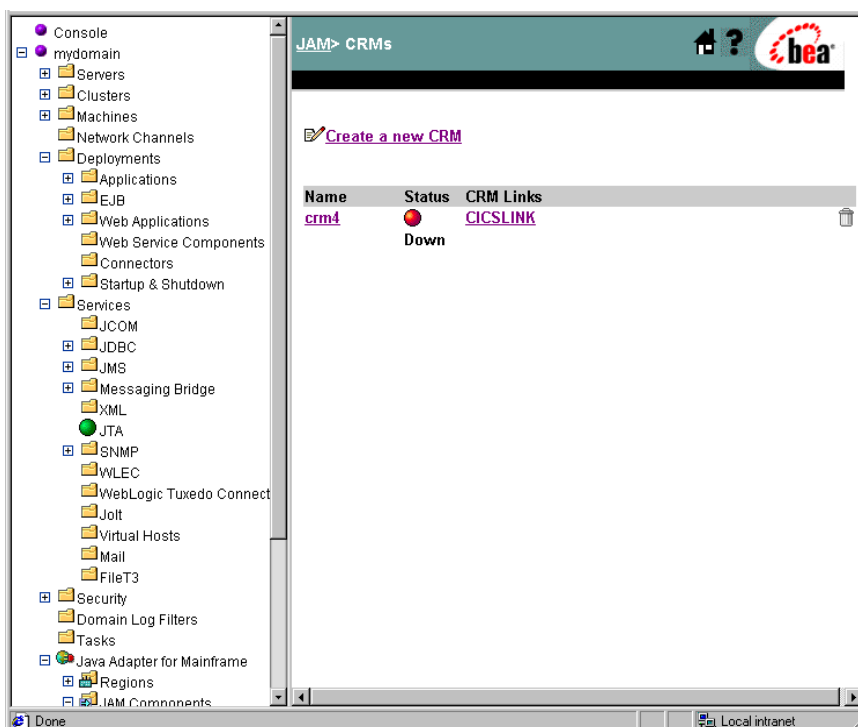
- [Listing CRM Definitions](#)
- [Editing a CRM Definition](#)
- [Deleting a CRM Definition](#)

### Listing CRM Definitions

To list a CRM definition, perform the following step:

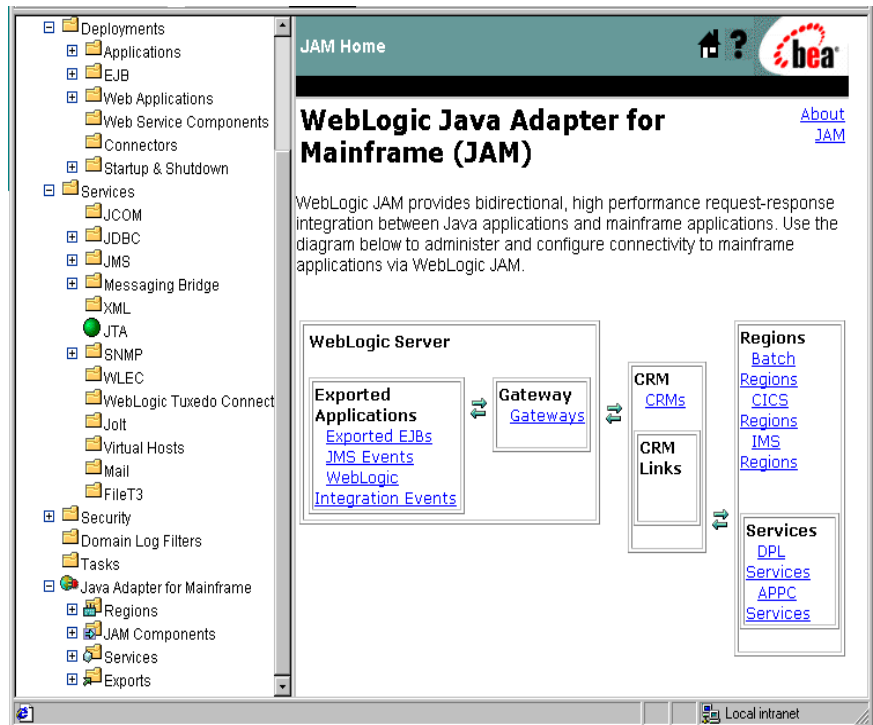


1. From the WebLogic JAM home page, click **CRM**. The List CRM page displays.

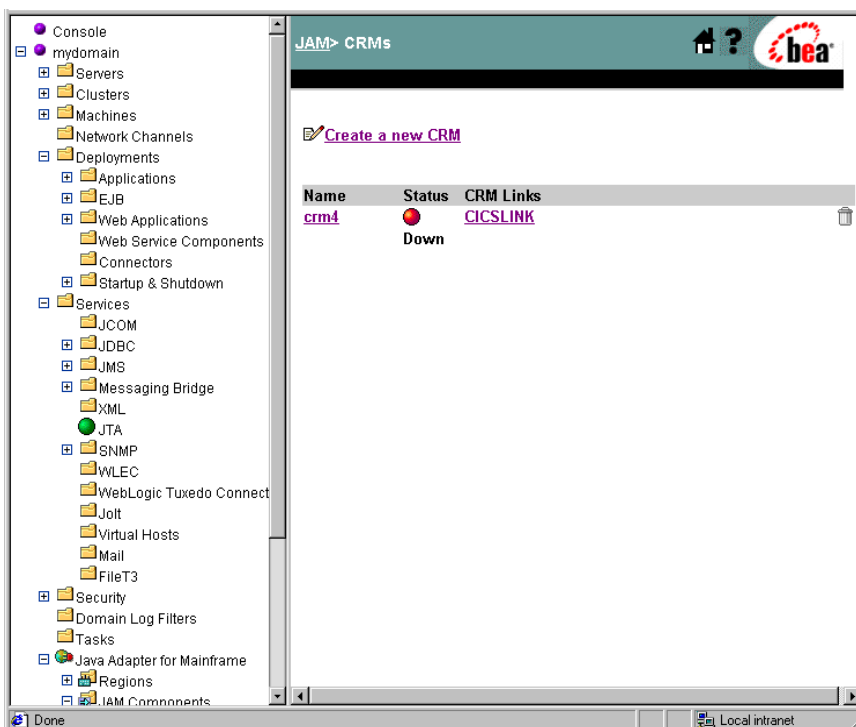


### Editing a CRM Definition

To edit a CRM definition, perform the following steps:

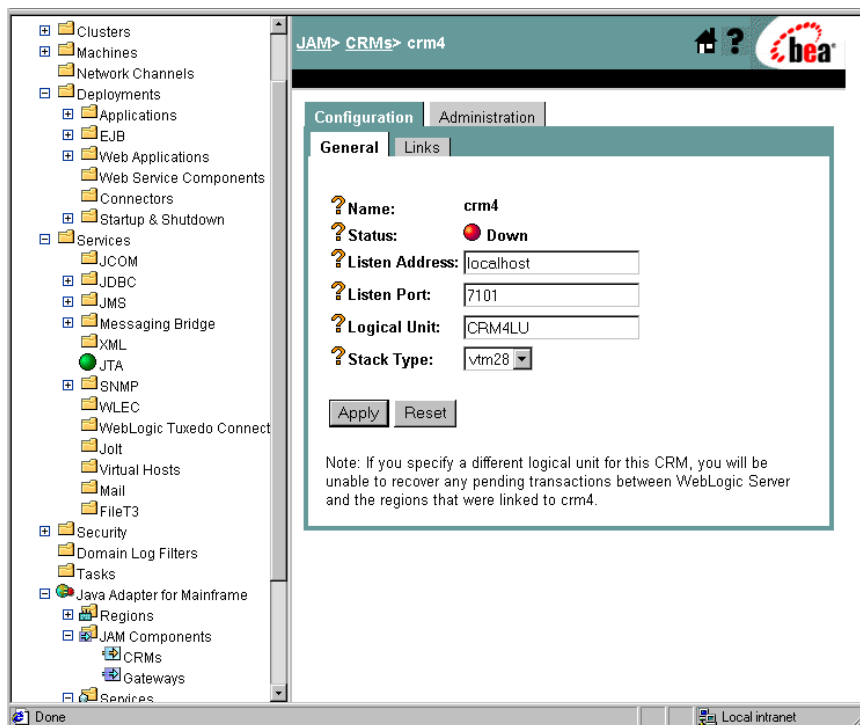


1. From the WebLogic JAM home page, click **CRMs**. The List CRM page displays.



## 5 Modifying Your Configuration

2. Click the CRM definition that you want to edit.



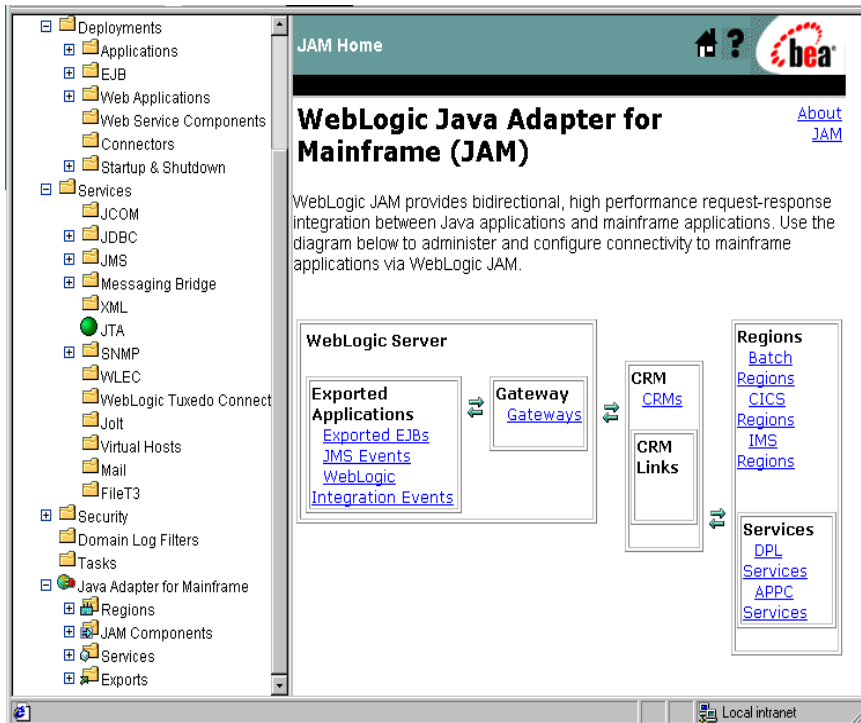
3. Change the CRM parameters to the appropriate values.
4. Click **Apply** to save your changes.

### Deleting a CRM Definition

You can delete a CRM definition by selecting the associated trash can icon in the WebLogic Administration Console. The trash can icon is enabled only when the CRM has no CRM Links linked to it and if no Gateway is assigned to it. If you try to delete a CRM that is referenced by a CRM Link or Gateway entity in the WebLogic JAM configuration, an error message displays, and the CRM is not deleted.

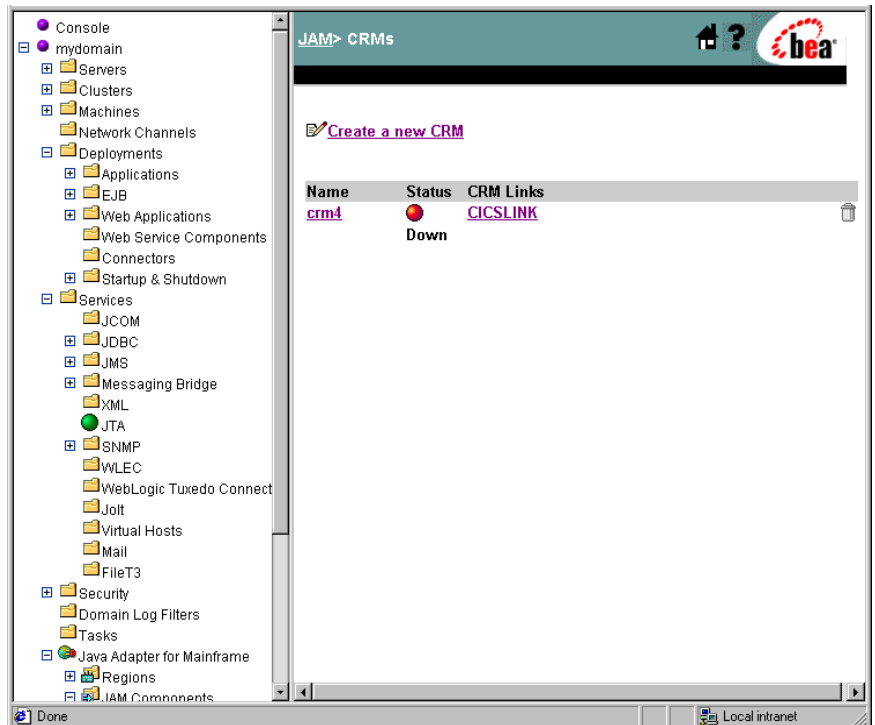


To delete a CRM definition, perform the following steps:



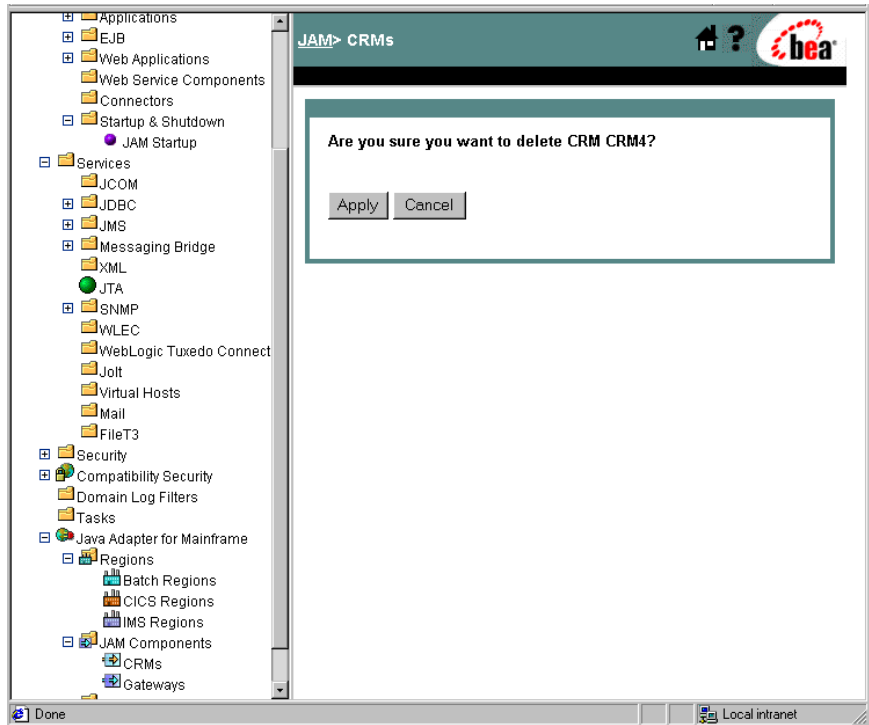
1. From the WebLogic JAM home page, click **CRMs**. The List CRM page displays.

## 5 Modifying Your Configuration



2. Make sure the associated CRM links are deleted. If the CRM Link is not stopped, see [“Starting and Stopping CRM Links”](#) for instructions on stopping a CRM Link.

3. Click the trash can icon.



4. Click **Apply** to finish deleting the CRM definition.

## Modifying CRM Link Definitions

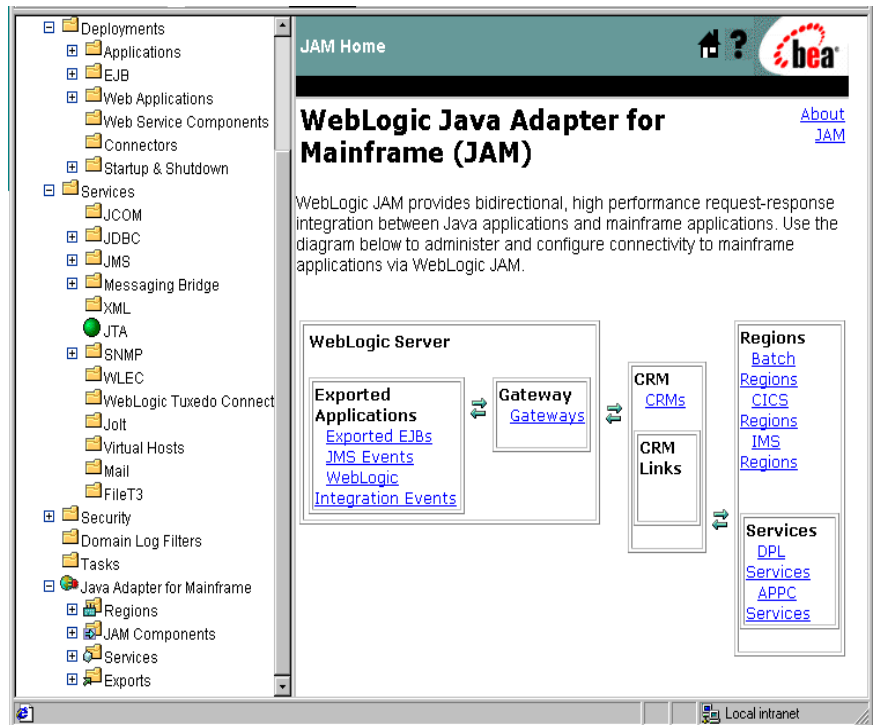
You can view and modify CRM Link definitions in the following ways:

- [Listing CRM Link Definitions](#)
- [Editing an Existing CRM Link](#)
- [Deleting a CRM Link](#)

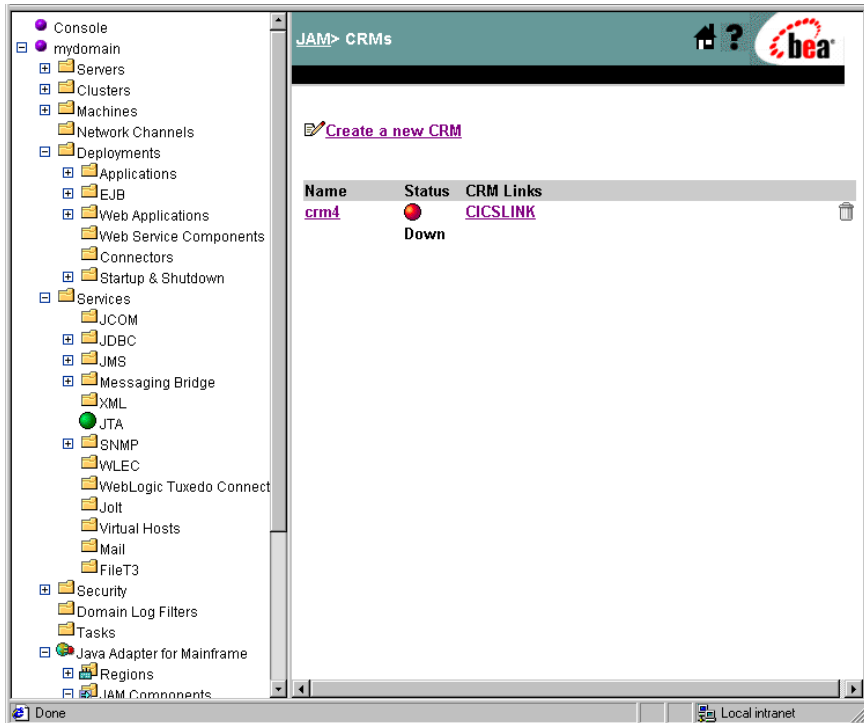
**Note:** All Gateways connected to a CRM must be restarted if link definitions are changed.

### Listing CRM Link Definitions

To list a CRM Link definition, perform the following steps:

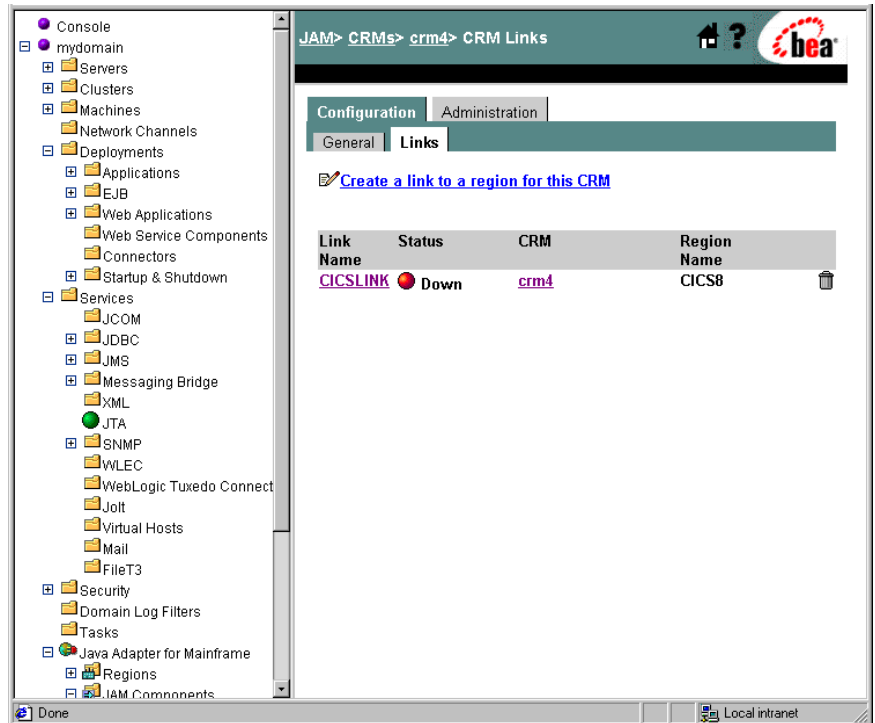


1. From the WebLogic JAM home page, click **CRMs**. The List CRM page displays.



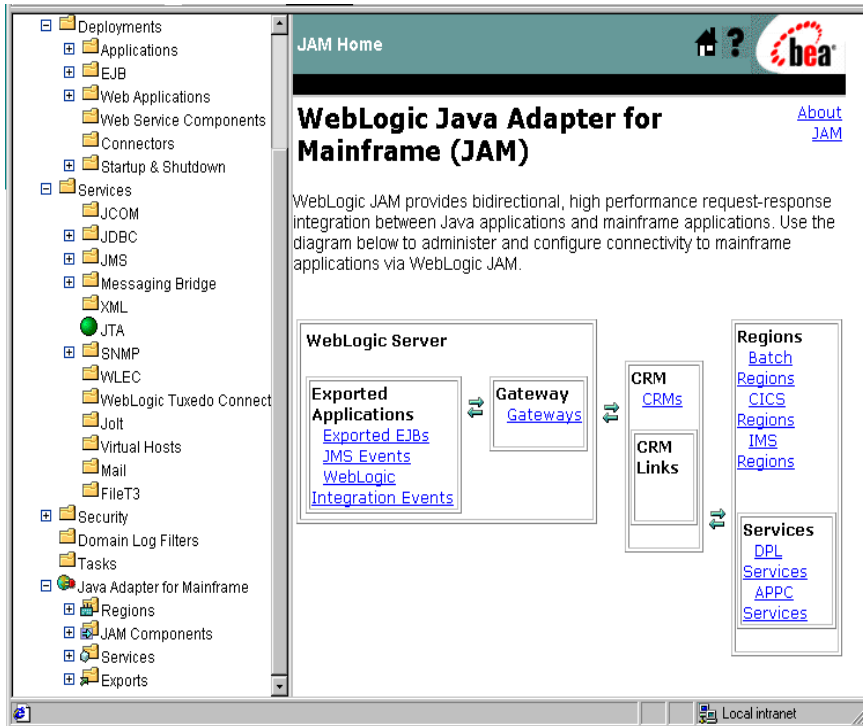
## 5 *Modifying Your Configuration*

2. Click the CRM Links you want to list.



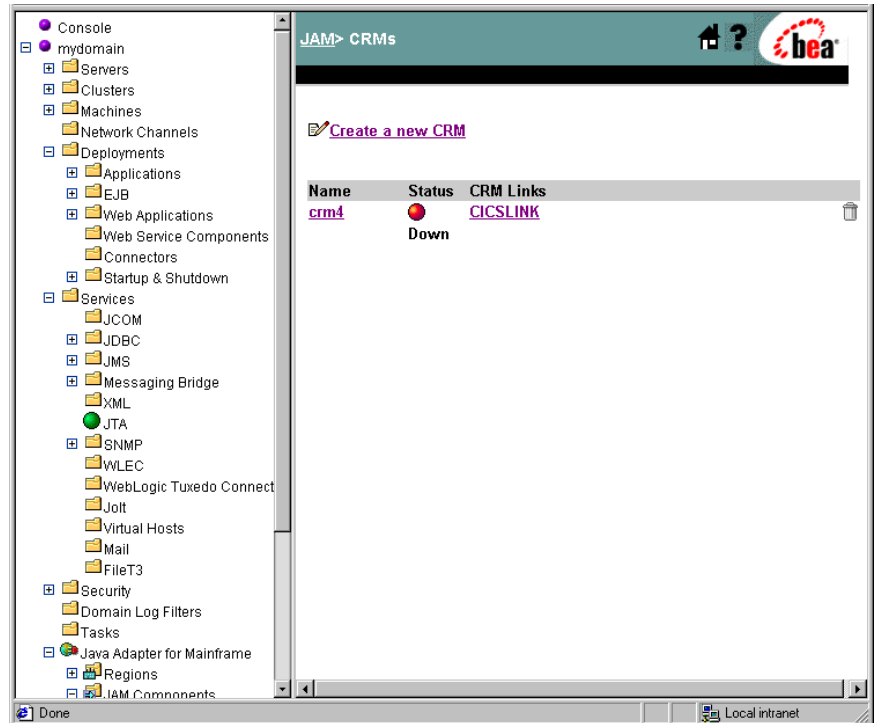
### Editing an Existing CRM Link

To edit an existing CRM link, perform the following steps:



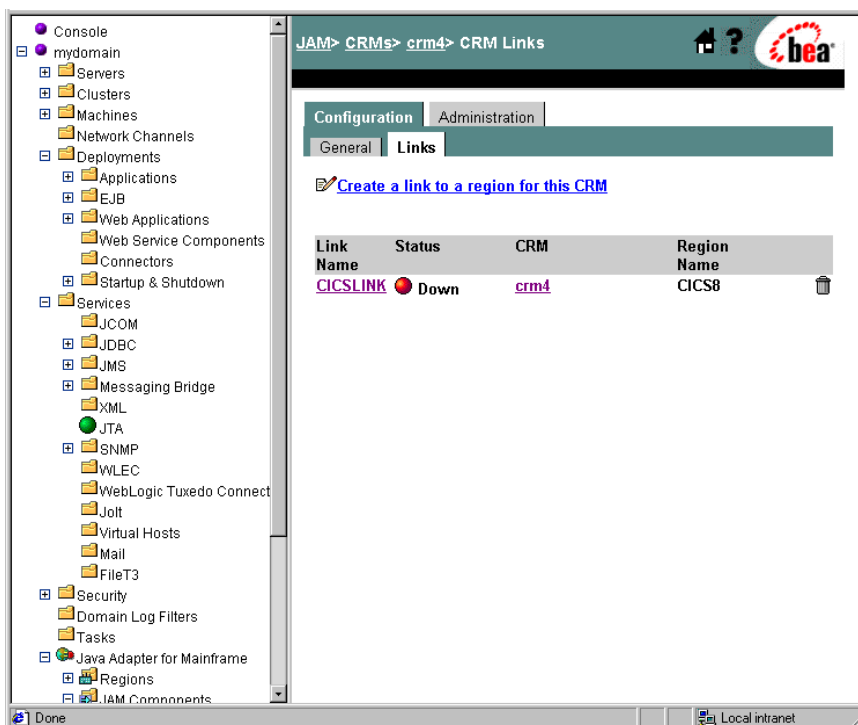
## 5 Modifying Your Configuration

1. From the WebLogic JAM home page, click **CRMs**. The List CRM page displays.

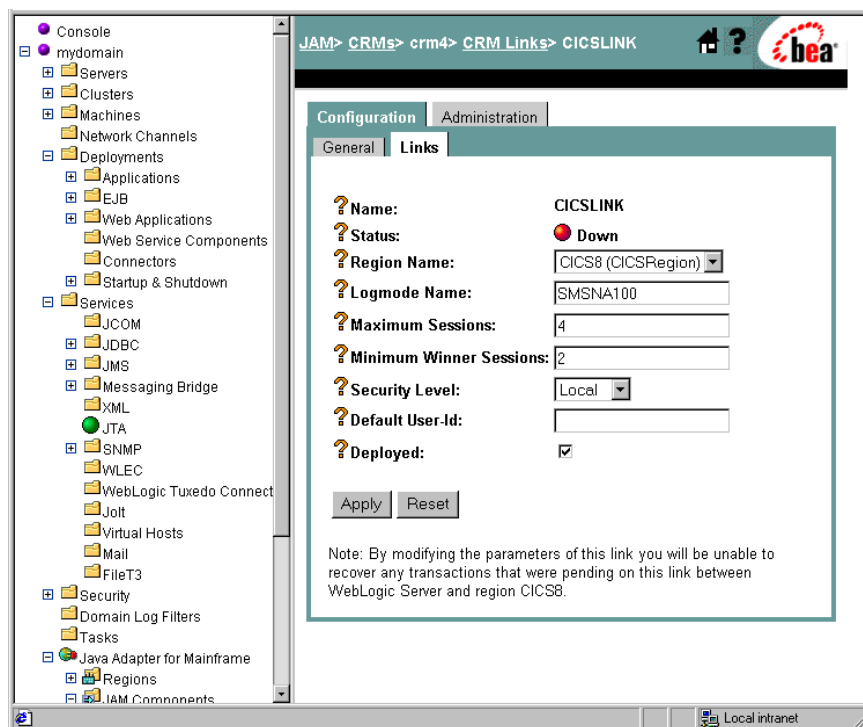




2. Click the CRM link you want to edit.



3. Click the CRM Link name.



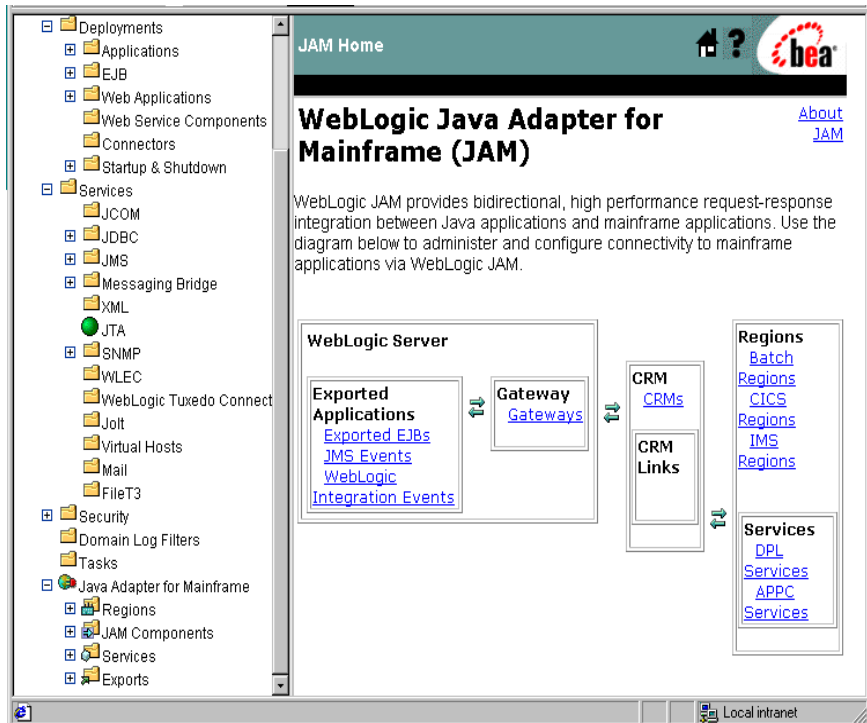
4. Change the CRM Link parameters to the appropriate values.

If the **Deployed** box is checked, the link will initialize when the Gateway starts. If the **Deployed** box is not checked, this link will not initialize or be available for use. The change to the **Deployed** box is applied when the Gateway is restarted.

5. Click **Apply** to save your changes.

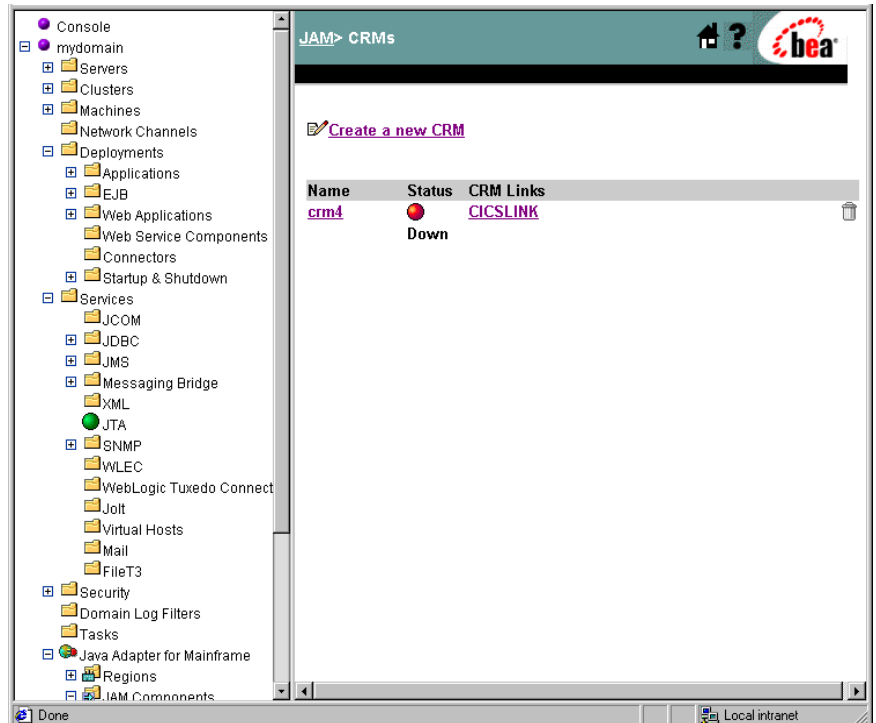
### Deleting a CRM Link

A CRM Link cannot be deleted if it is active. To delete a CRM Link, perform the following steps:

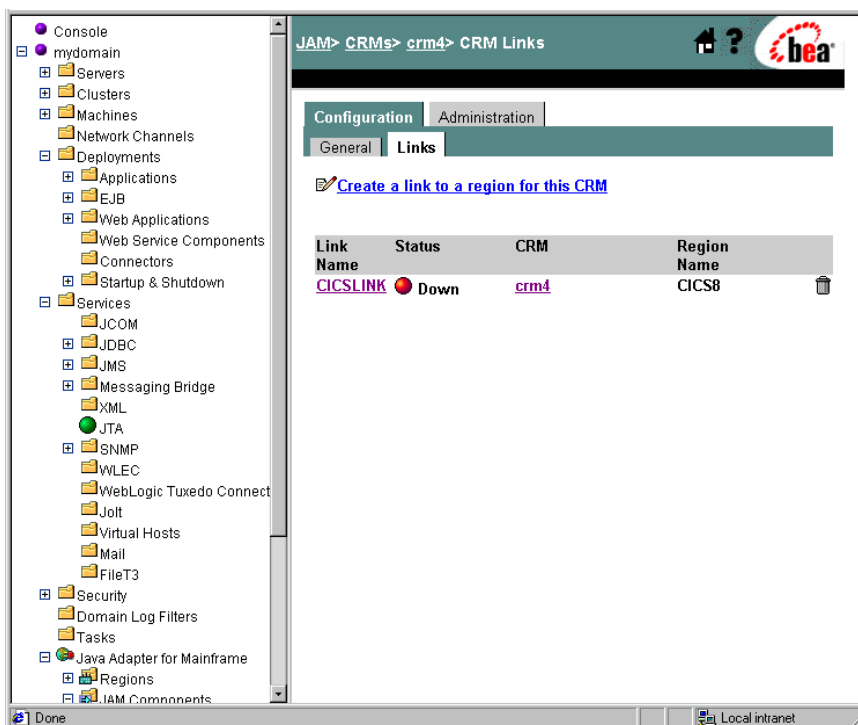


## 5 Modifying Your Configuration

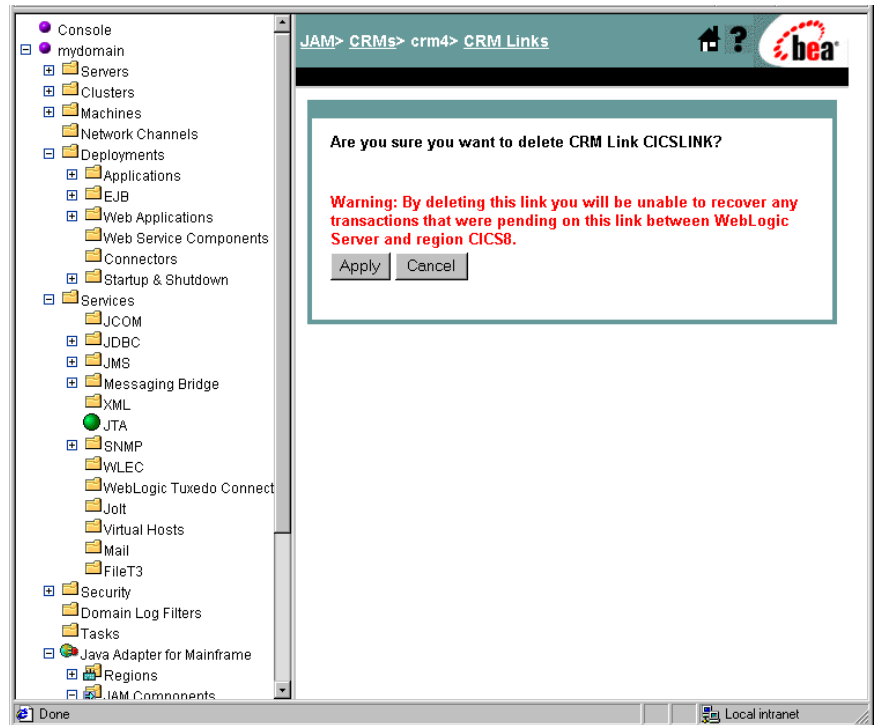
1. From the WebLogic JAM home page, click **CRMs**. The List CRM page displays.



2. Click the CRM Link you want to delete.



3. Click the Trash Can icon for the CRM Link you want to delete.



Click **Apply** to finish deleting the CRM Link.

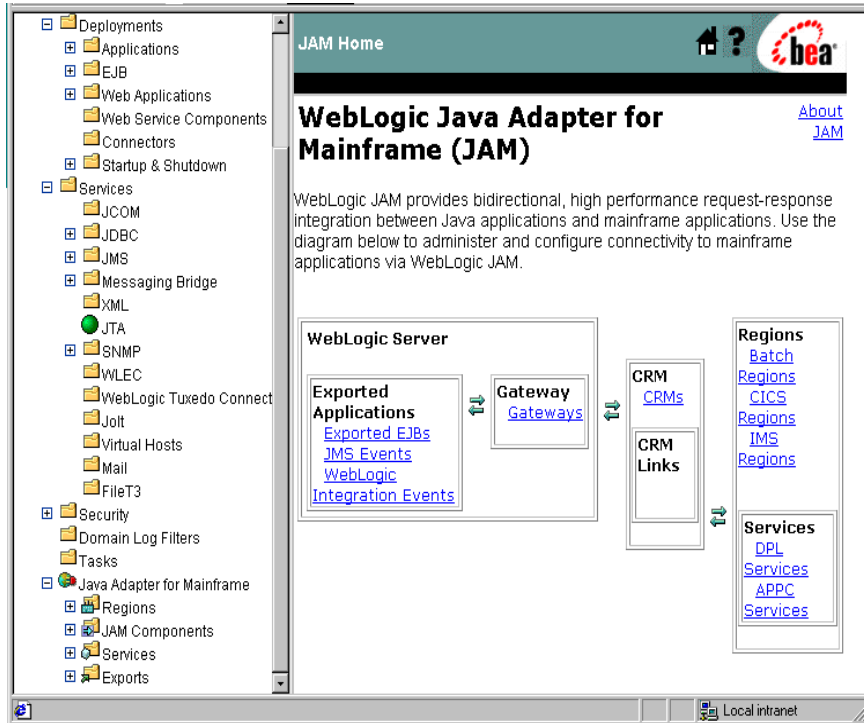
## Modifying Gateway Definitions

You can view and modify Gateway definitions in the following ways:

- [Listing Gateway Definitions](#)
- [Editing a Gateway Definition](#)
- [Deleting a Gateway Definition](#)

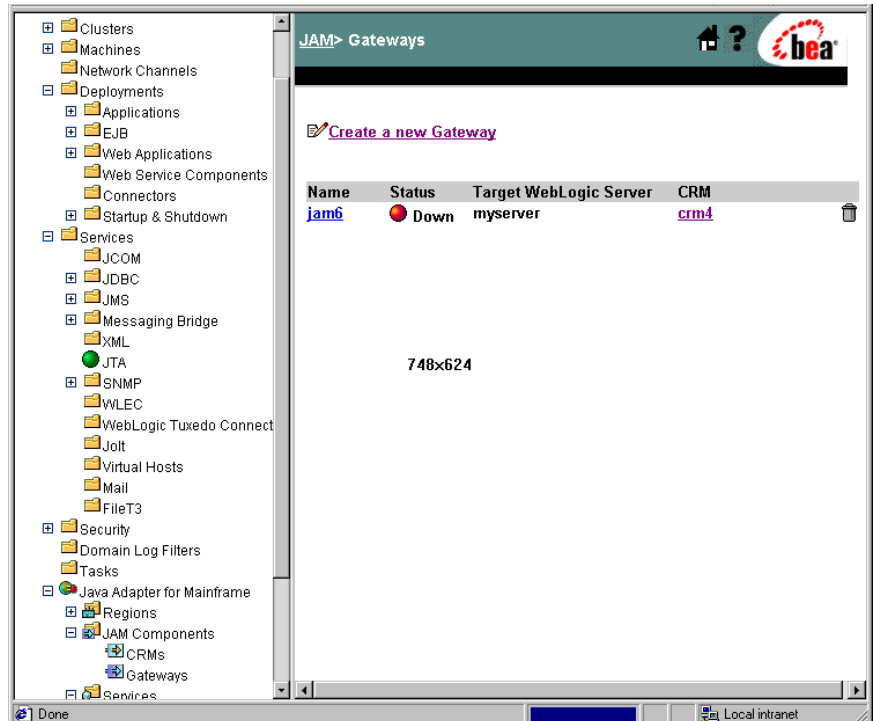
## Listing Gateway Definitions

To list existing Gateways, perform the following step:



## 5 *Modifying Your Configuration*

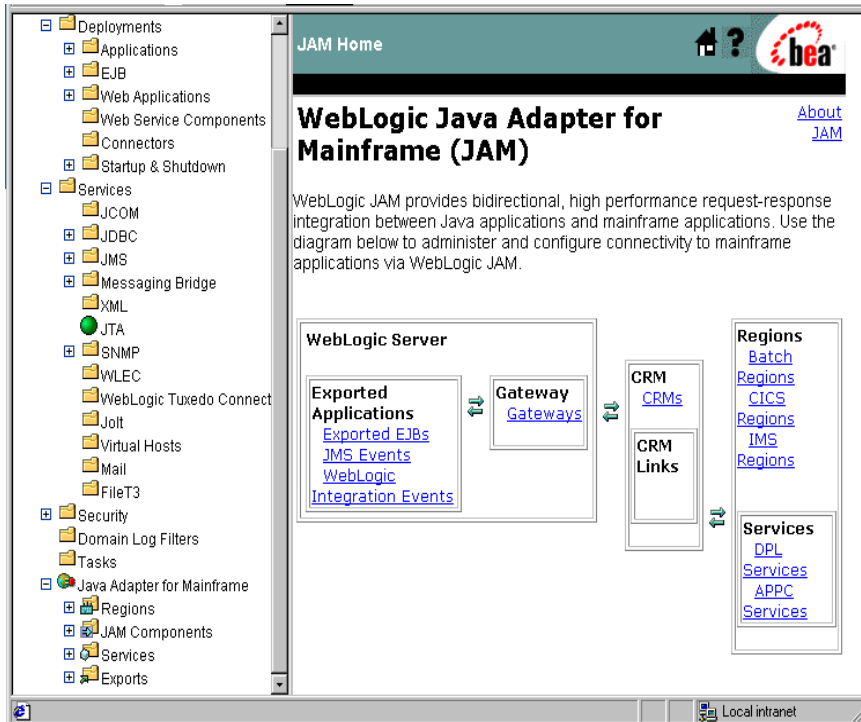
1. From the WebLogic JAM home page, click **Gateways**. The List Gateways page displays.





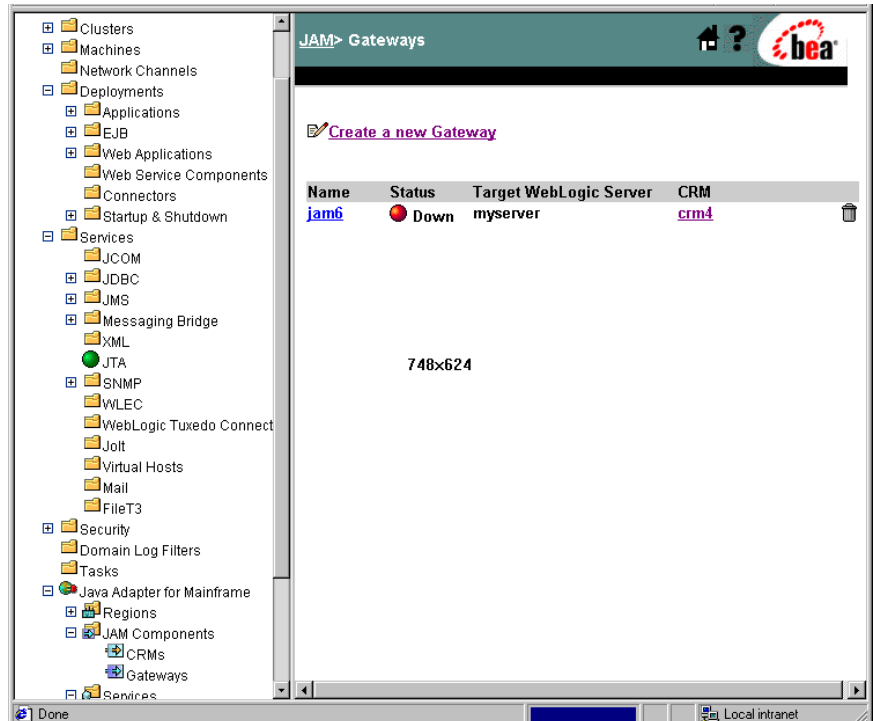
### Editing a Gateway Definition

To edit a Gateway definition, perform the following steps:



## 5 Modifying Your Configuration

1. From the WebLogic JAM home page, click **Gateways**. The List Gateways page displays.



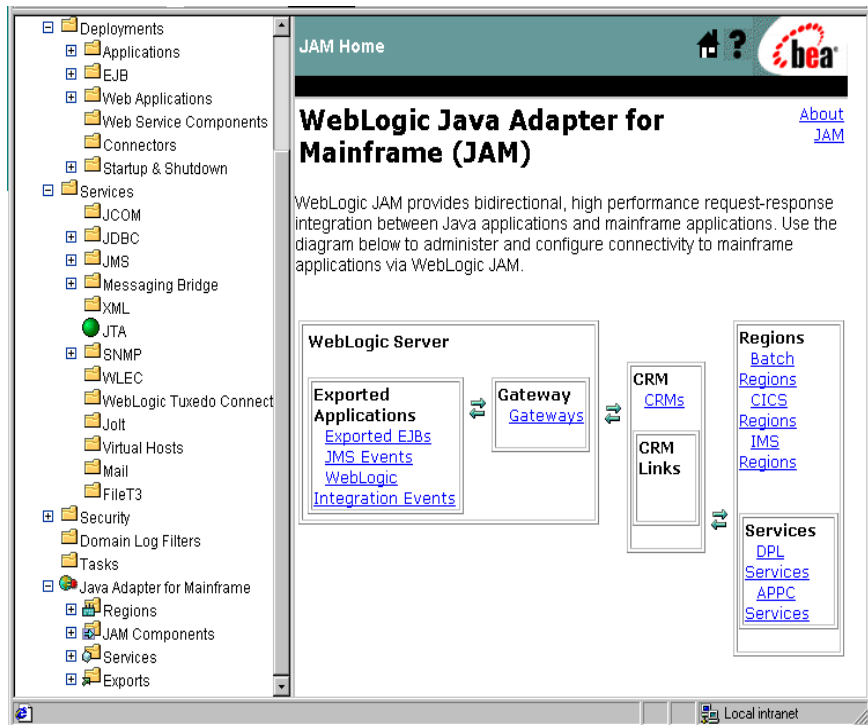
2. Click the Gateway definition that you want to edit.



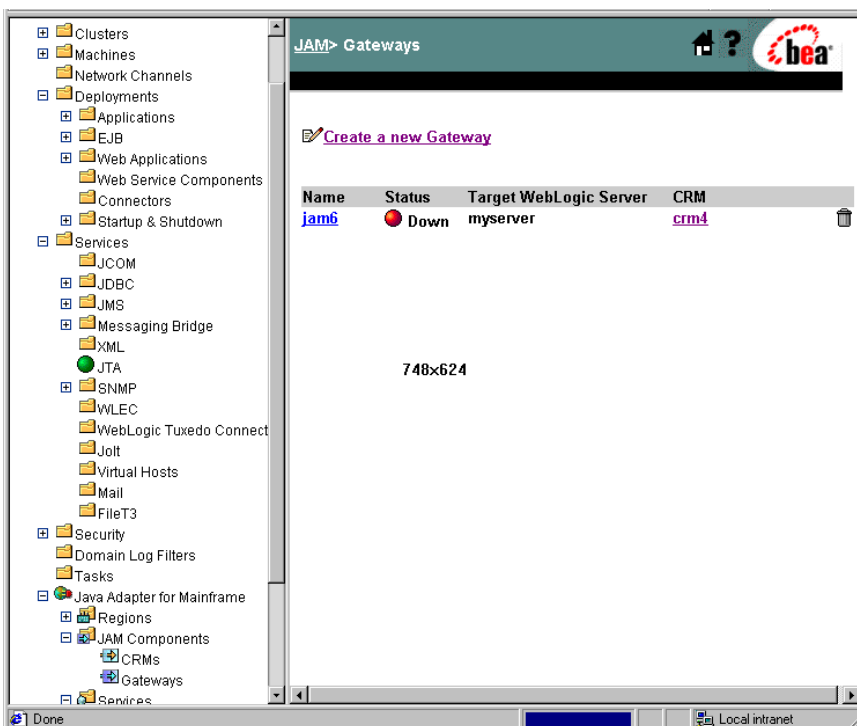
3. Change the Gateway parameters to the appropriate values.
4. Click **Apply** to save your changes.

### Deleting a Gateway Definition

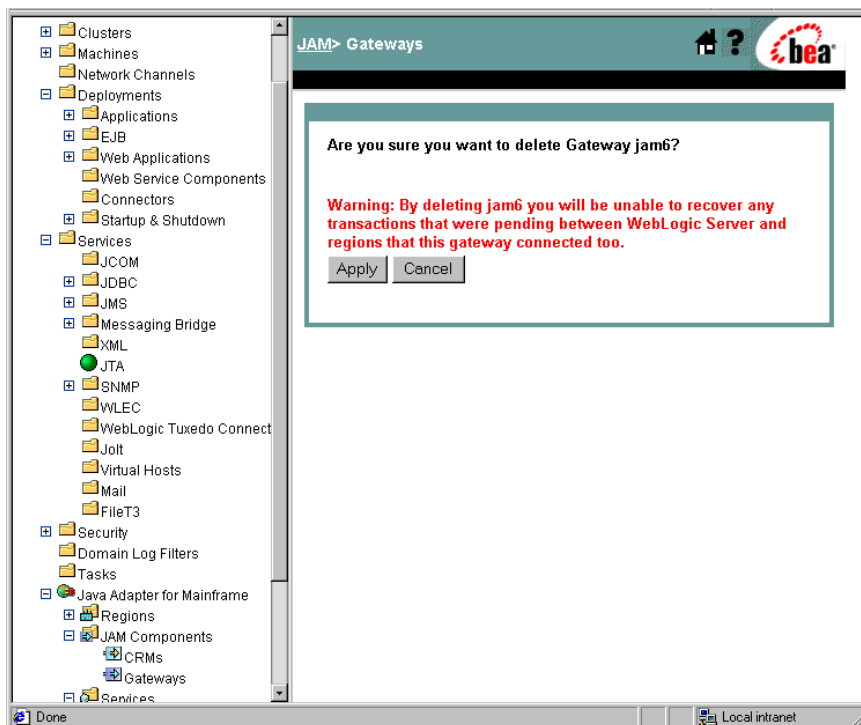
A Gateway cannot be deleted if it is active. To delete a Gateway definition, perform the following steps:



1. From the WebLogic JAM home page, click **Gateways**. The List Gateways page displays.



2. Click the trash can icon.



3. Click **Apply** to finish deleting the Gateway definition.

# Modifying Your Application Integration Configuration

The WebLogic Administration Console provides you with the ability to modify your application integration configurations. You have the following options:

- [Modifying a DPL Service Definition](#)
- [Modifying an APPC Service Definition](#)

- [Modifying an Exported EJB Definition](#)
- [Modifying a JMS Event Definition](#)
- [Modifying a WebLogic Integration Event Definition](#)

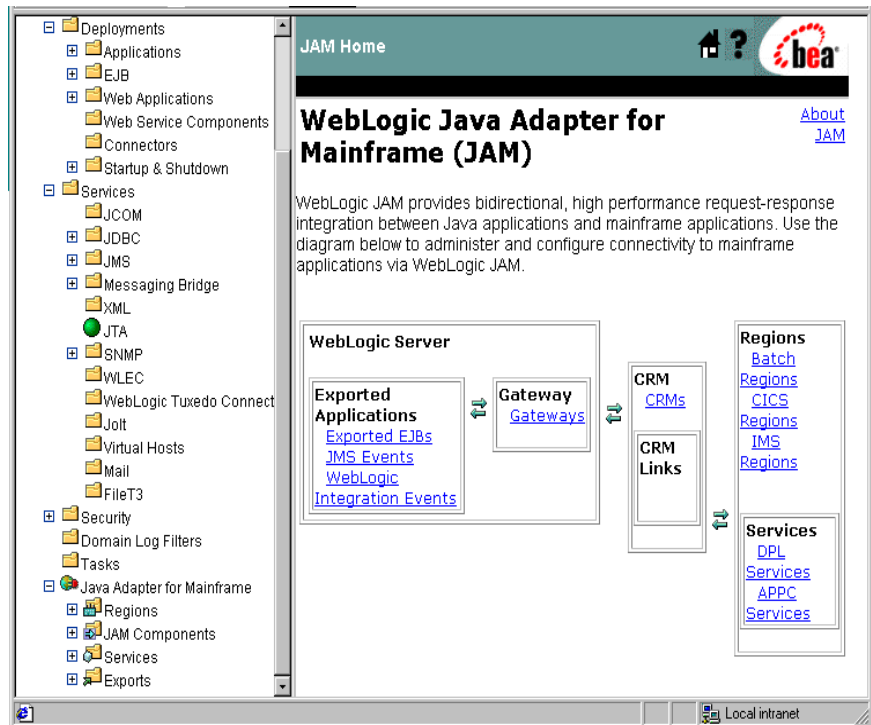
## **Modifying a DPL Service Definition**

To modify a DPL service definition, you have the following options:

- [Editing a DPL Service Definition](#)
- [Deleting a DPL Service Definition](#)

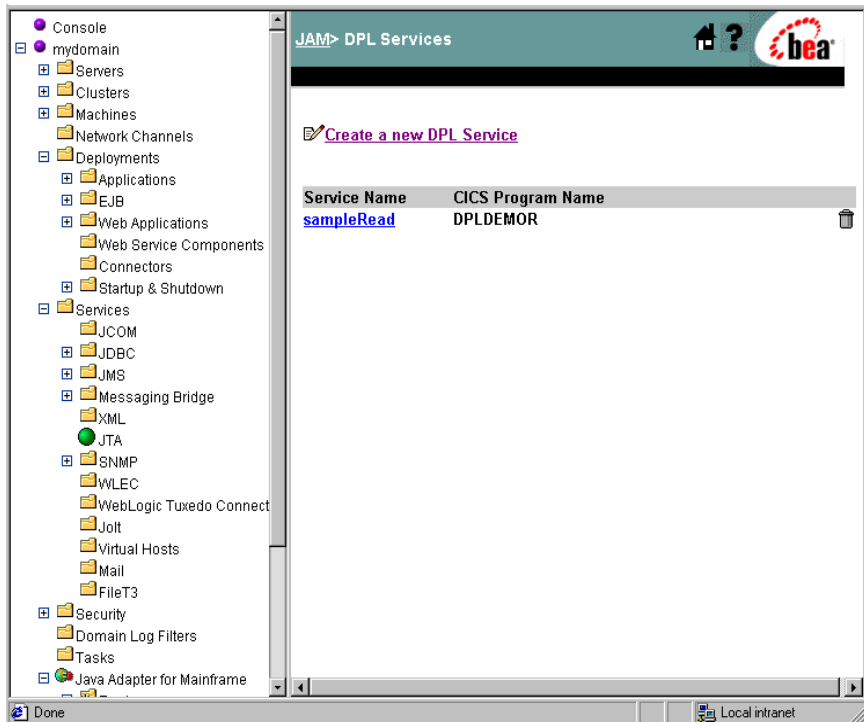
### Editing a DPL Service Definition

To edit a DPL service definition, perform the following steps:

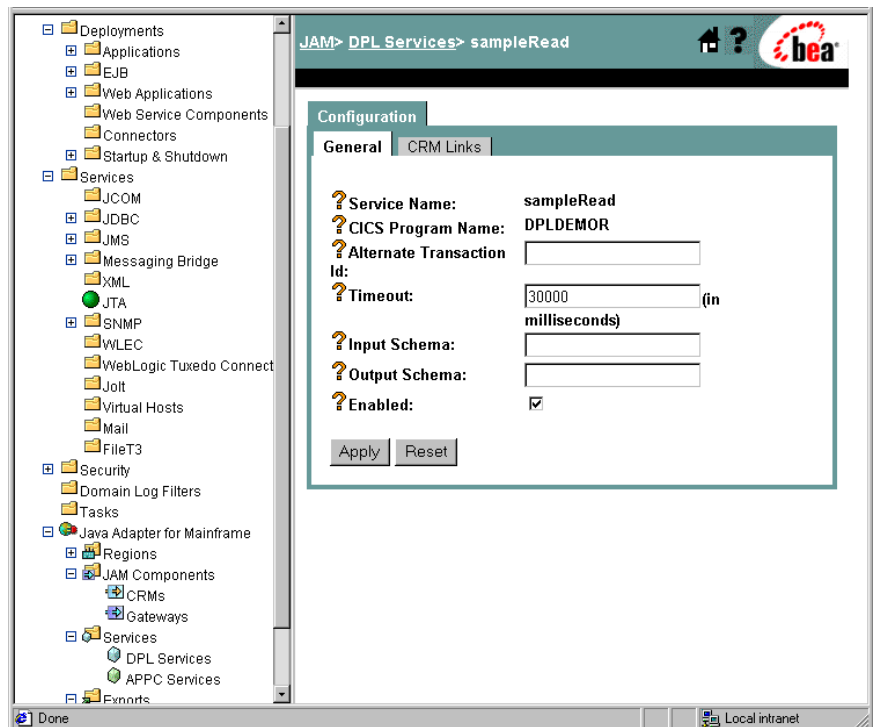




1. From the WebLogic JAM home page, click **DPL Services**. The List DPL Services page displays.



2. Click the DPL Service definition that you want to edit.



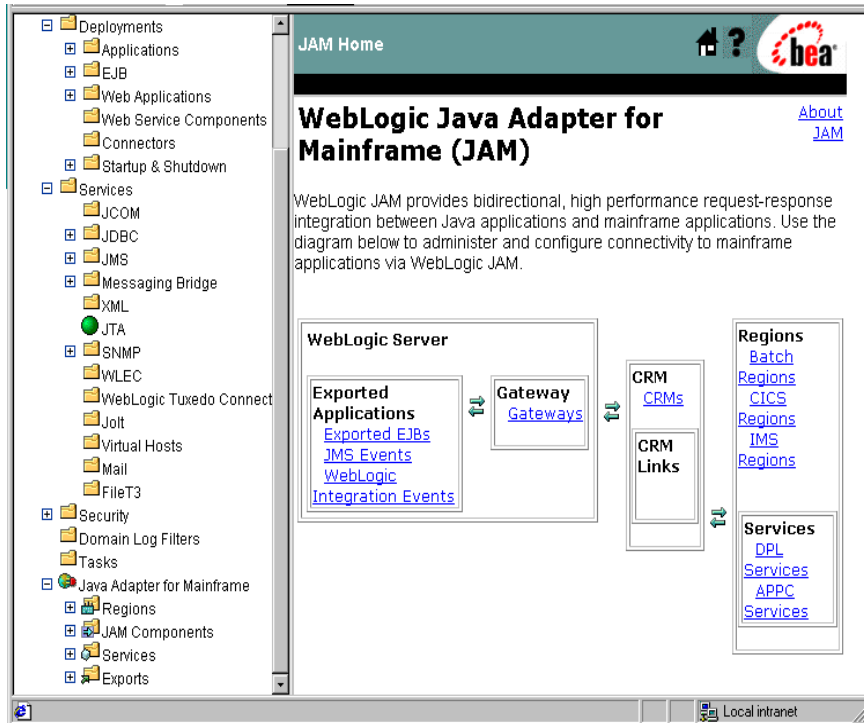
3. Change the DPL Service parameters to the appropriate values.

A check in the **Enabled** box makes the service available for use. Removing the check from the **Enabled** box disables the service.

4. Click **Apply** to save your changes.

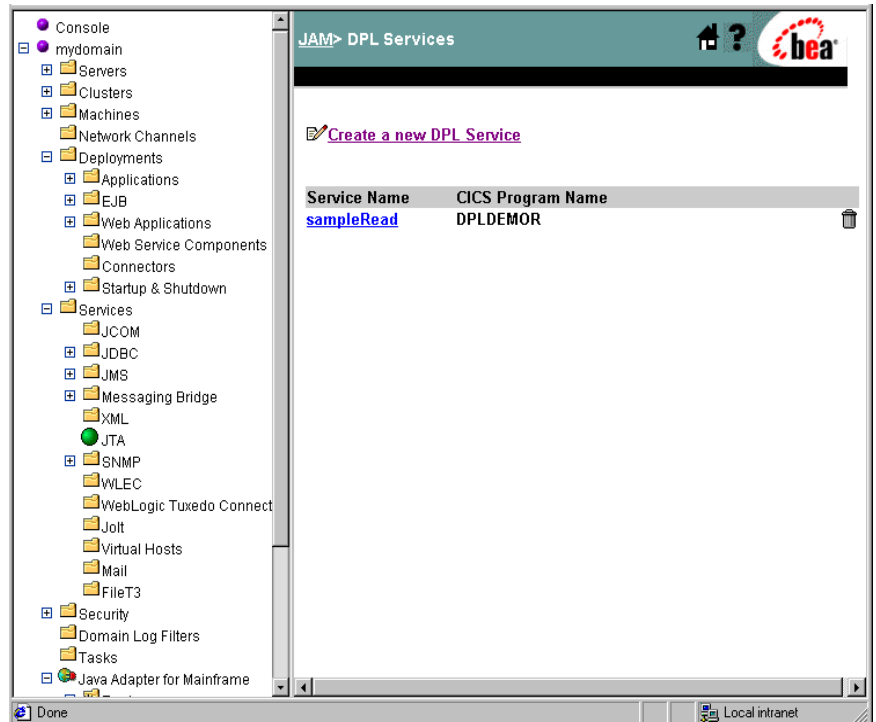
### Deleting a DPL Service Definition

To delete a DPL Service definition, perform the following steps:

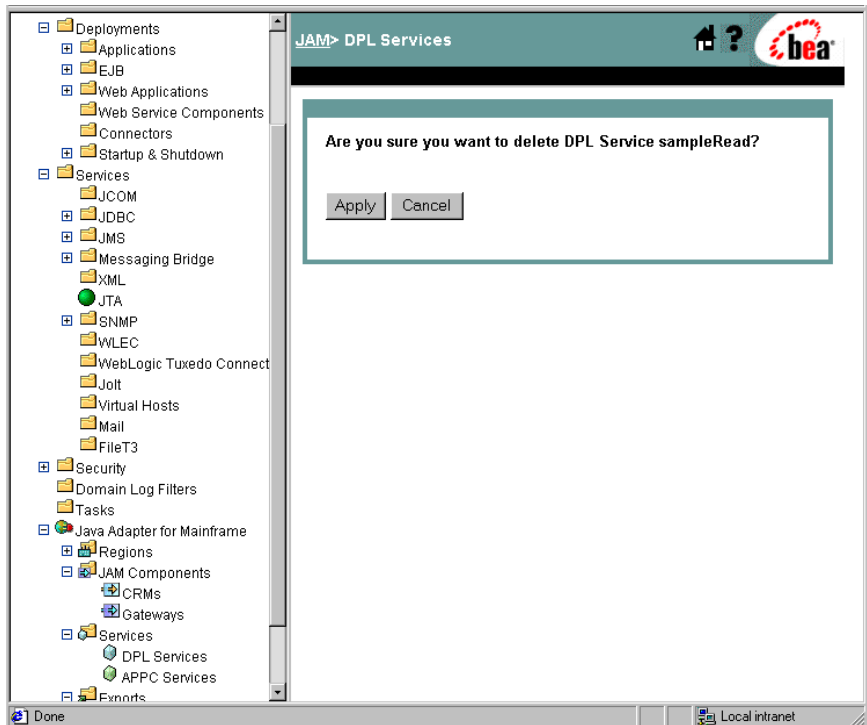


## 5 *Modifying Your Configuration*

1. From the WebLogic JAM home page, click **DPL Services**. The List DPL Services page displays.



2. Click the trash can icon.



3. Click **Apply** to finish deleting the DPL Service definition.

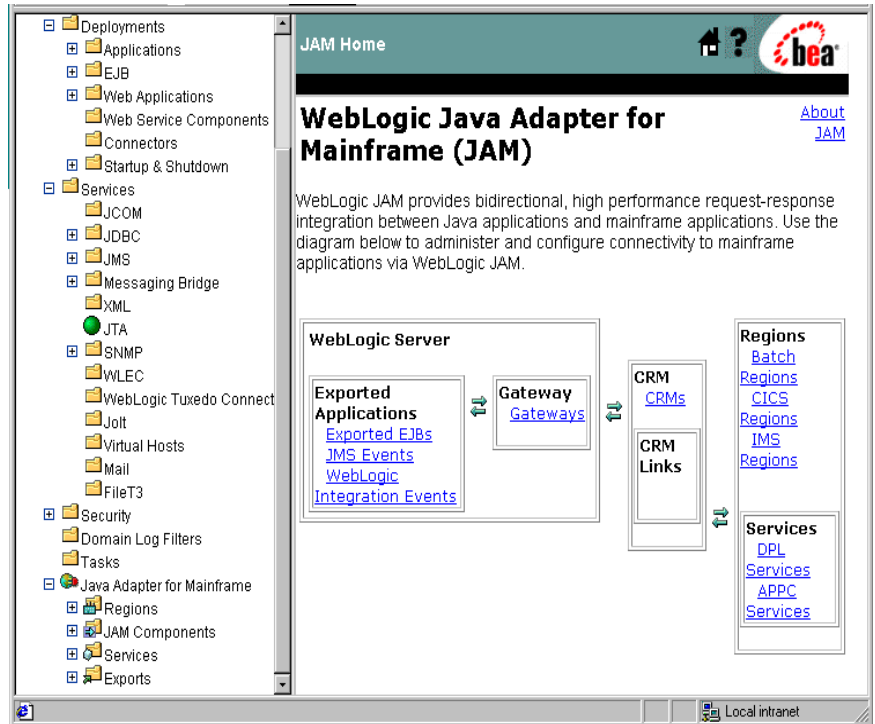
## Modifying an APPC Service Definition

To modify an APPC service definition, you have the following options:

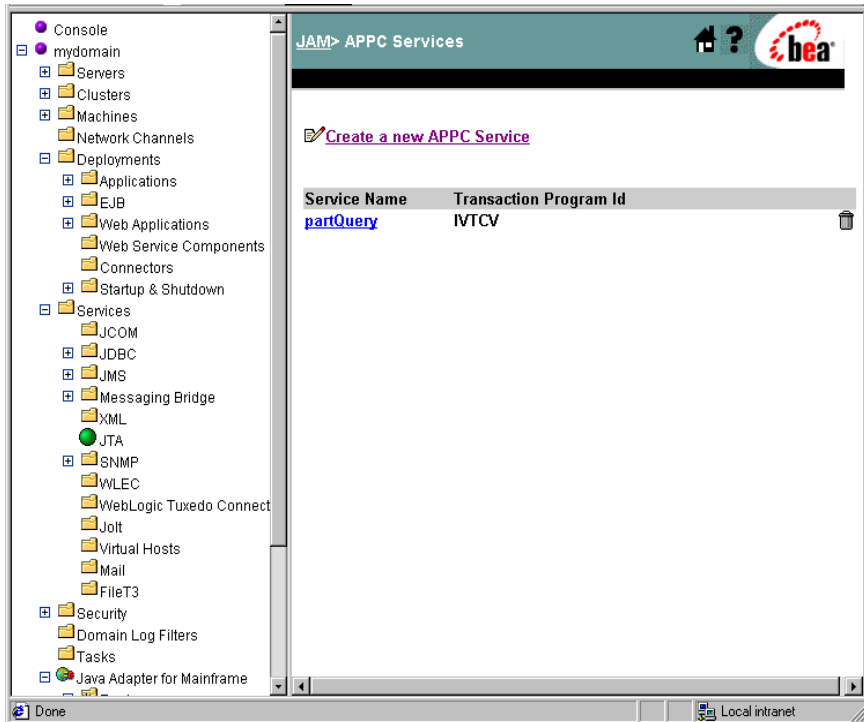
- [Editing an APPC Service Definition](#)
- [Deleting an APPC Service Definition](#)

### Editing an APPC Service Definition

To edit an APPC definition, perform the following steps:

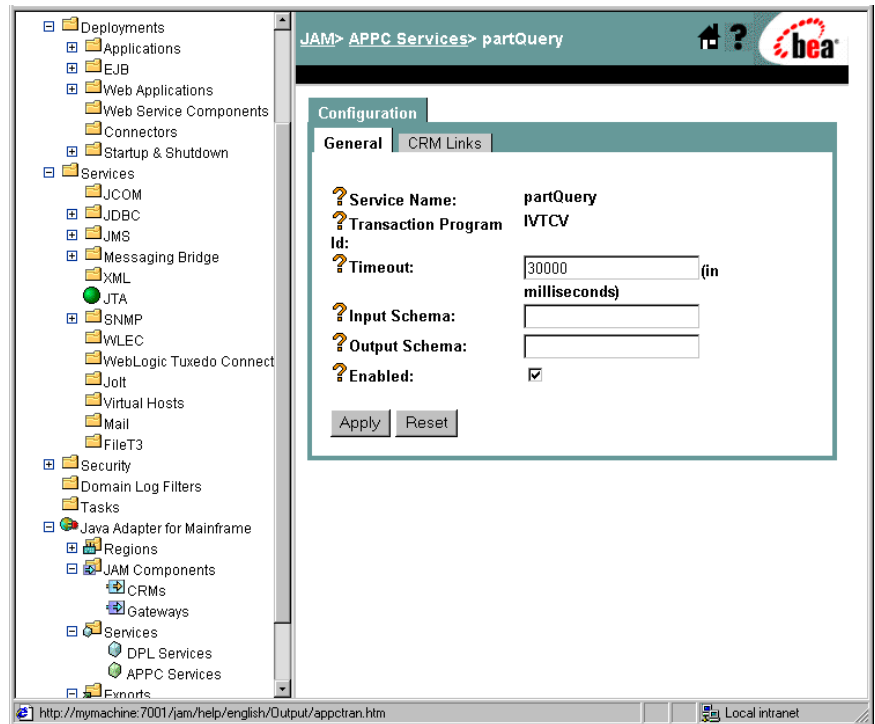


1. From the WebLogic JAM home page, click **APPC Services**. The List APPC Services page displays.



## 5 Modifying Your Configuration

2. Click the APPC Service definition that you want to edit.



3. Change the APPC Service parameters to the appropriate values.

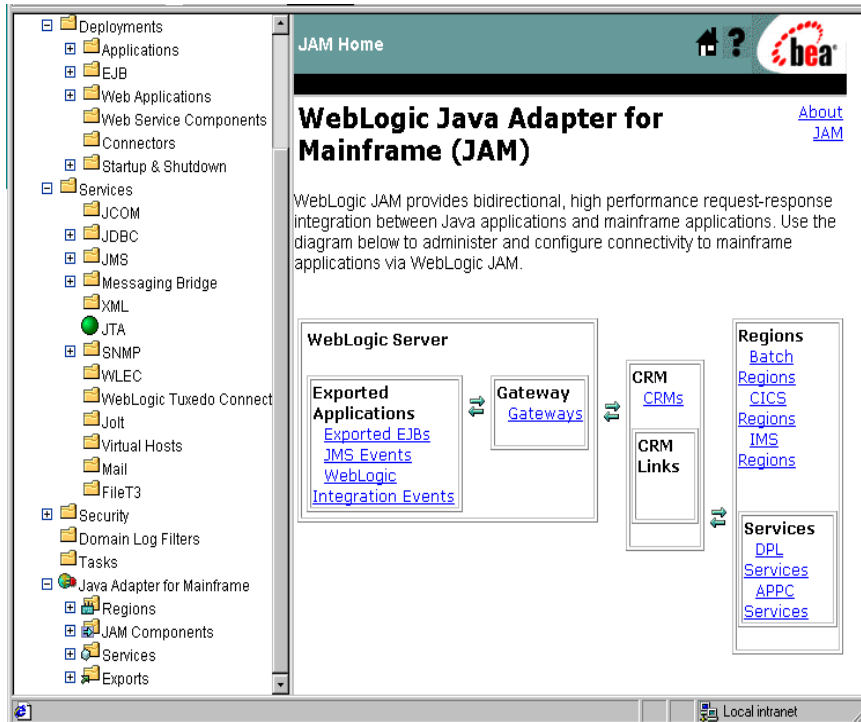
A check in the **Enabled** box makes the service available for use. Removing the check from the **Enabled** box disables the service.

4. Click **Apply** to save your changes.



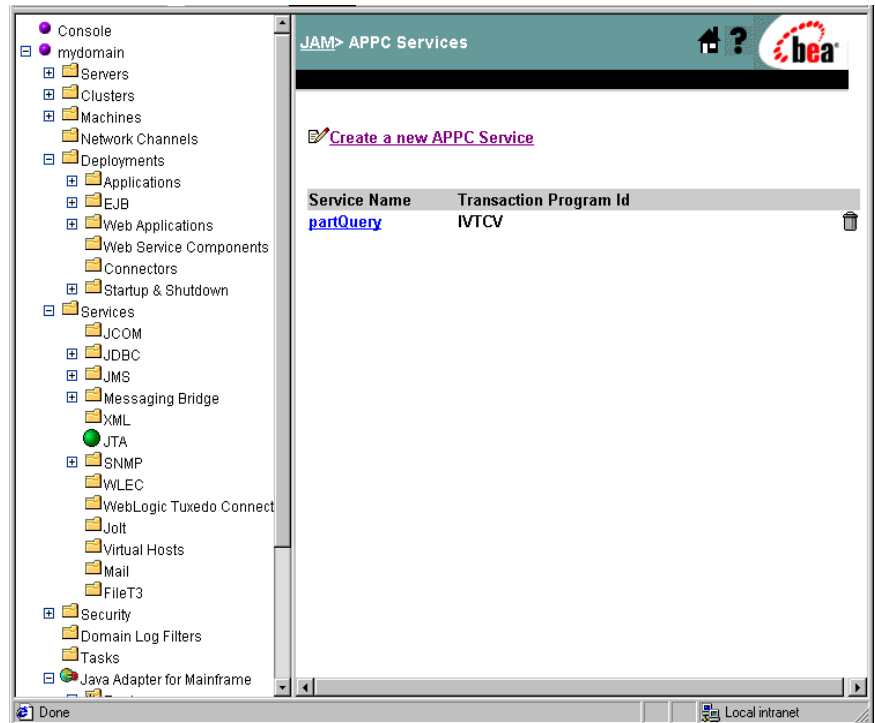
### Deleting an APPC Service Definition

To delete a APPC Service definition, perform the following steps:

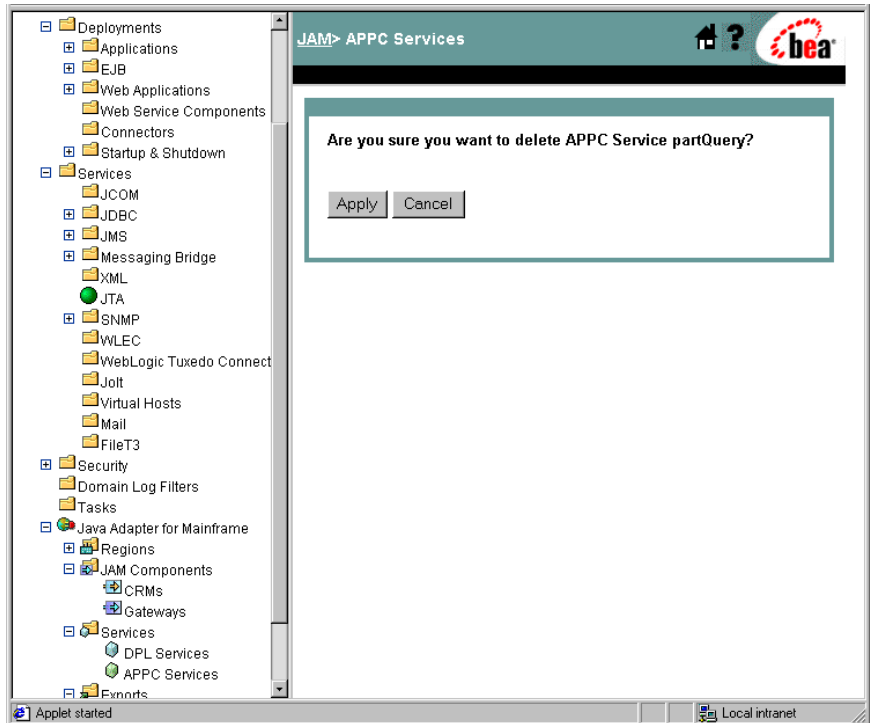


## 5 *Modifying Your Configuration*

1. From the WebLogic JAM home page, click **APPC Services**. The List APPC Services page displays.



2. Click the trash can icon.



3. Click **Apply** to finish deleting the APPC Service definition.

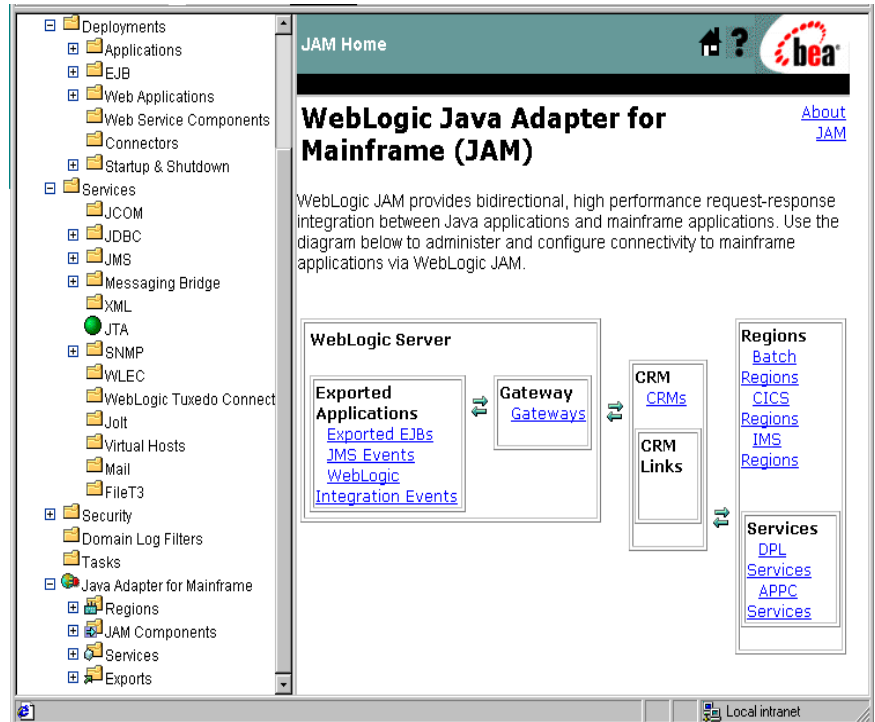
## Modifying an Exported EJB Definition

To modify an exported EJB definition, you have the following options:

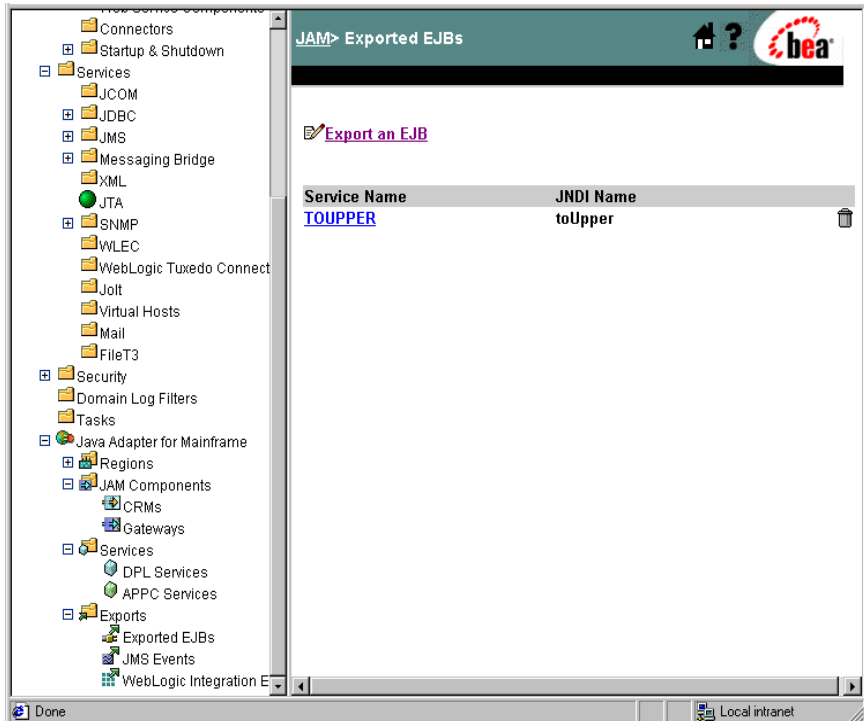
- [Editing an Exported EJB Definition](#)
- [Deleting an Exported EJB Definition](#)

### Editing an Exported EJB Definition

To edit an exported EJB, perform the following steps:

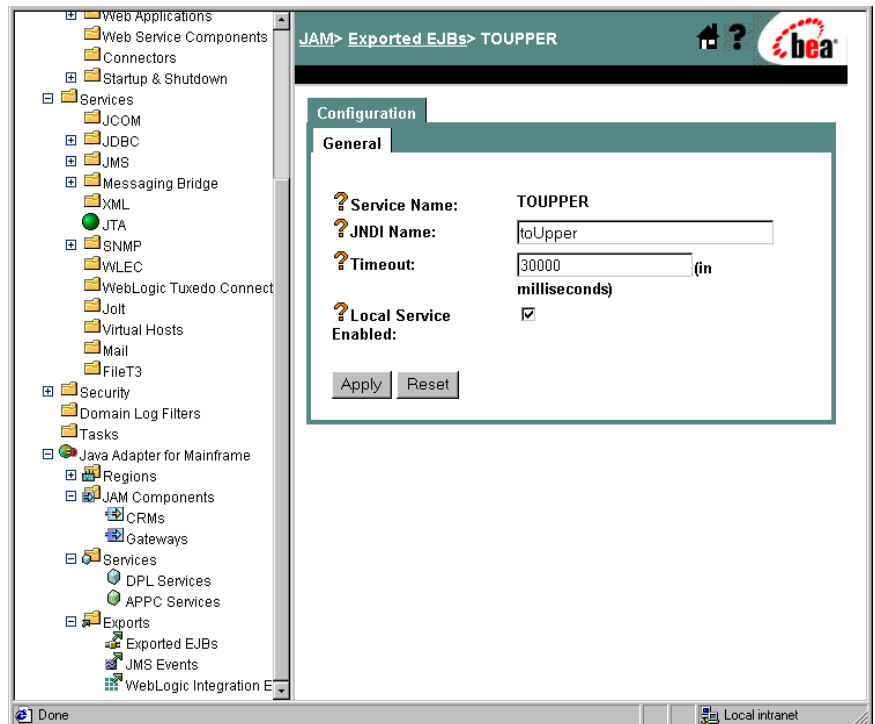


1. From the WebLogic JAM home page, click exported **EJBs**. The List Exported EJBs page displays.



## 5 Modifying Your Configuration

2. Click the EJB that you want to edit.



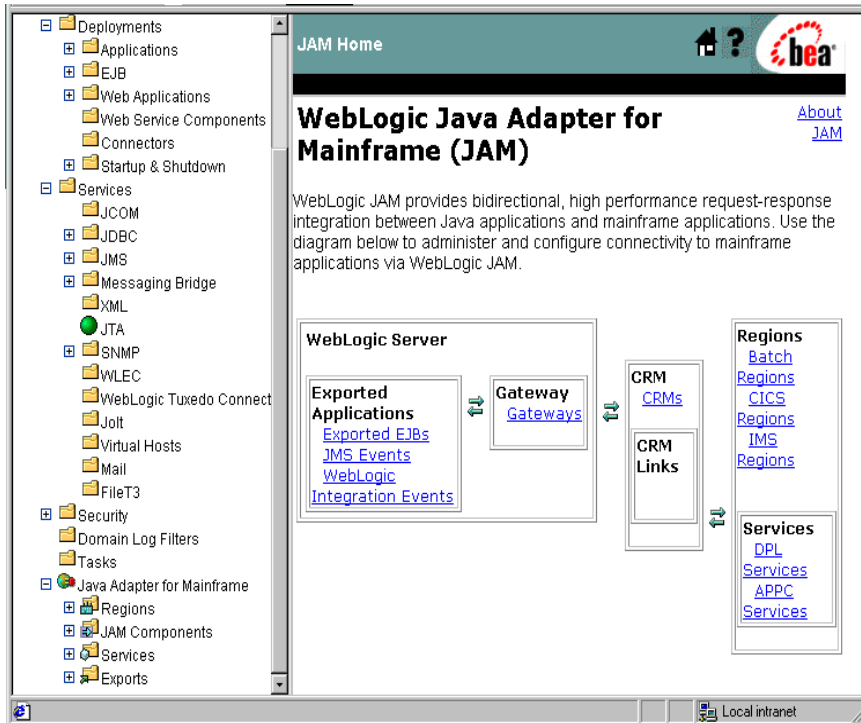
3. Change the EJB parameters to the appropriate values.

A check in the **Local Service Enabled** box makes the service available for use. Removing the check from the **Local Service Enabled** box disables the service.

4. Click **Apply** to save your changes.

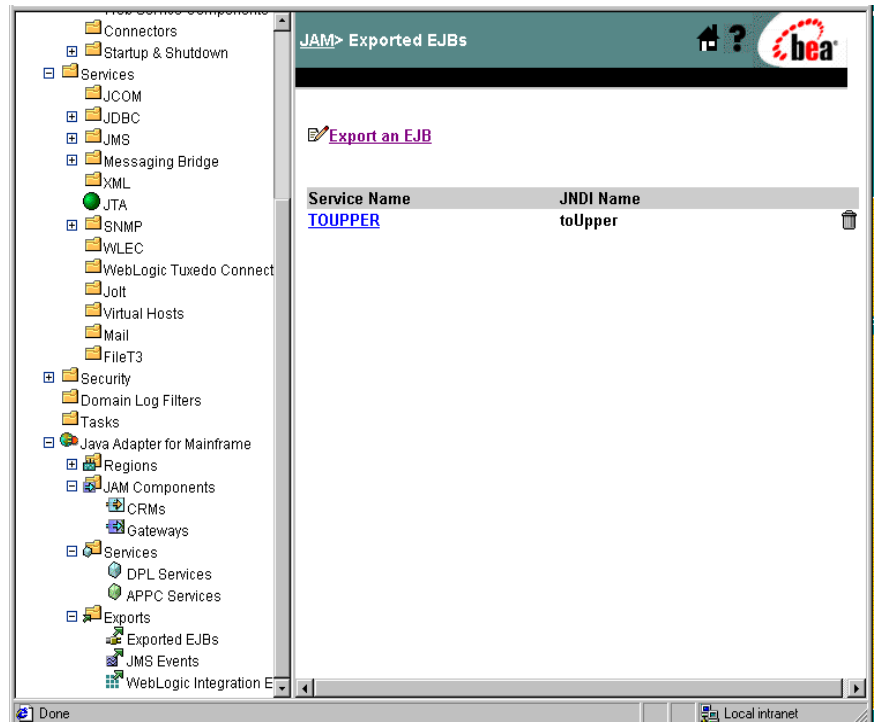
### Deleting an Exported EJB Definition

To delete an exported EJB, perform the following steps:



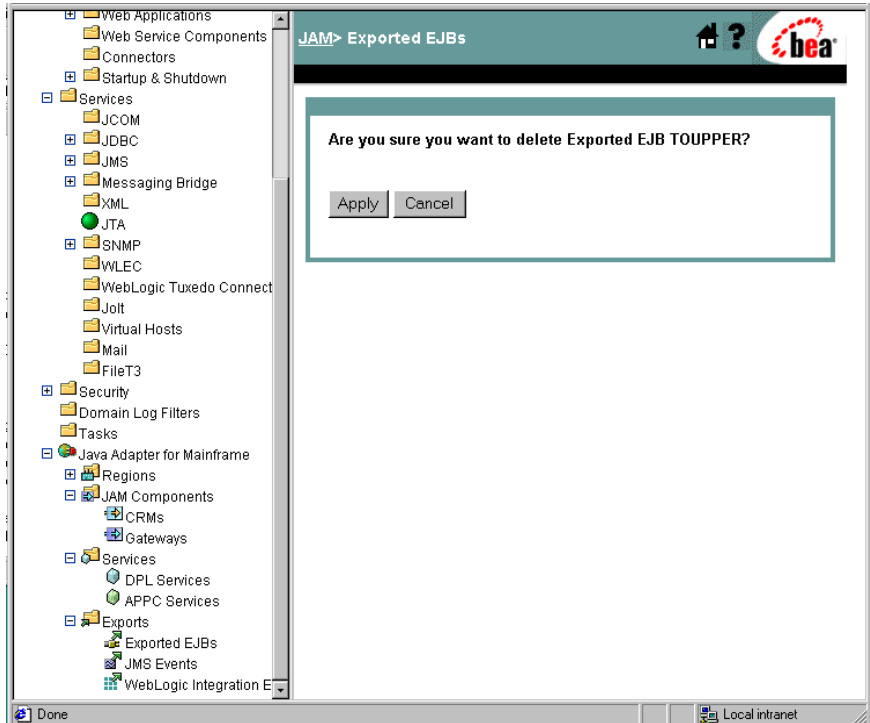
## 5 *Modifying Your Configuration*

1. From the WebLogic JAM home page, click **Exported EJBs**. The List Exported EJBs page displays.





2. Click the trash can icon.



3. Click **Apply** to finish deleting the EJB.

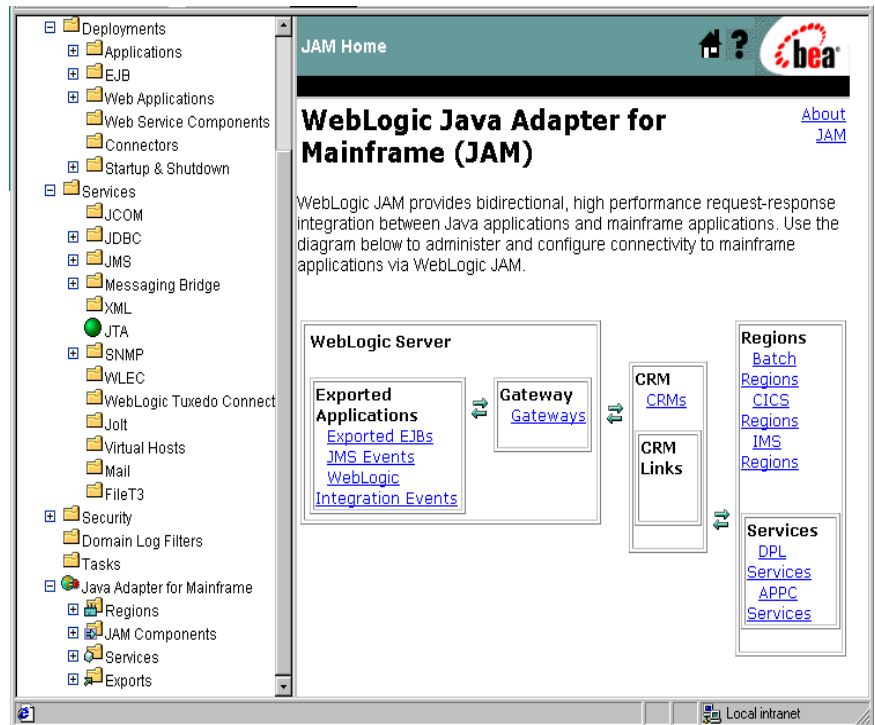
## Modifying a JMS Event Definition

To modify a JMS Event definition, you have the following options:

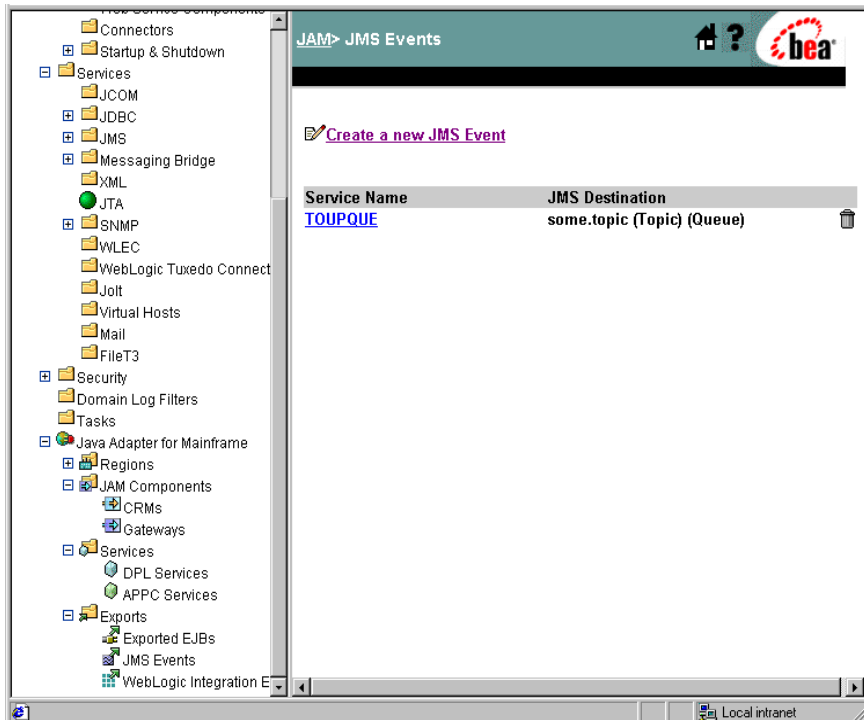
- [Editing a JMS Event](#)
- [Deleting a JMS Event](#)

### Editing a JMS Event

To edit a JMS Event, perform the following steps:

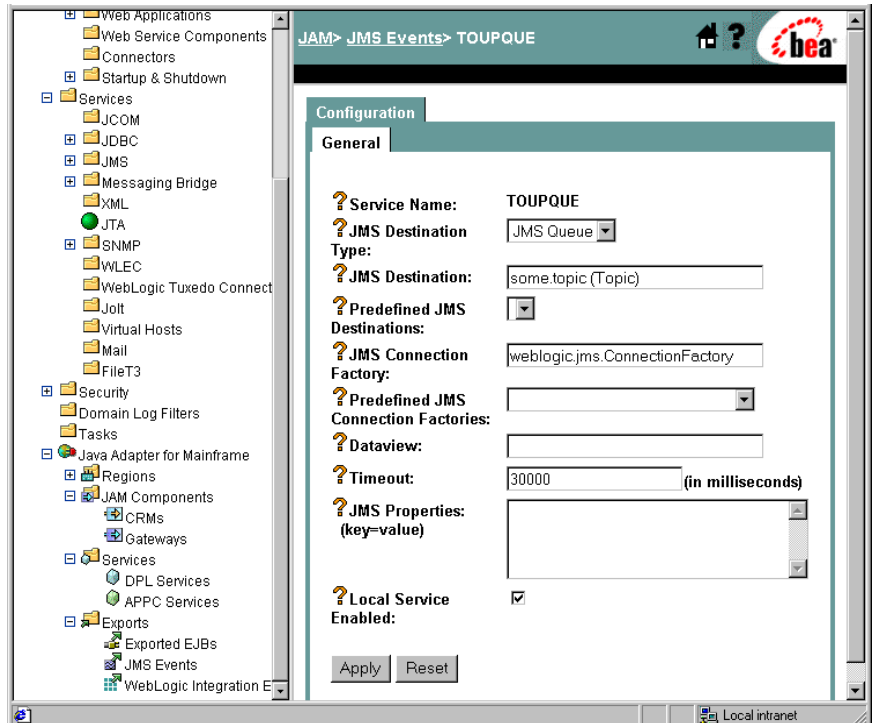


1. From the WebLogic JAM home page, click **JMS Events**. The List JMS Events page displays.



## 5 Modifying Your Configuration

2. Click the JMS Event that you want to edit.



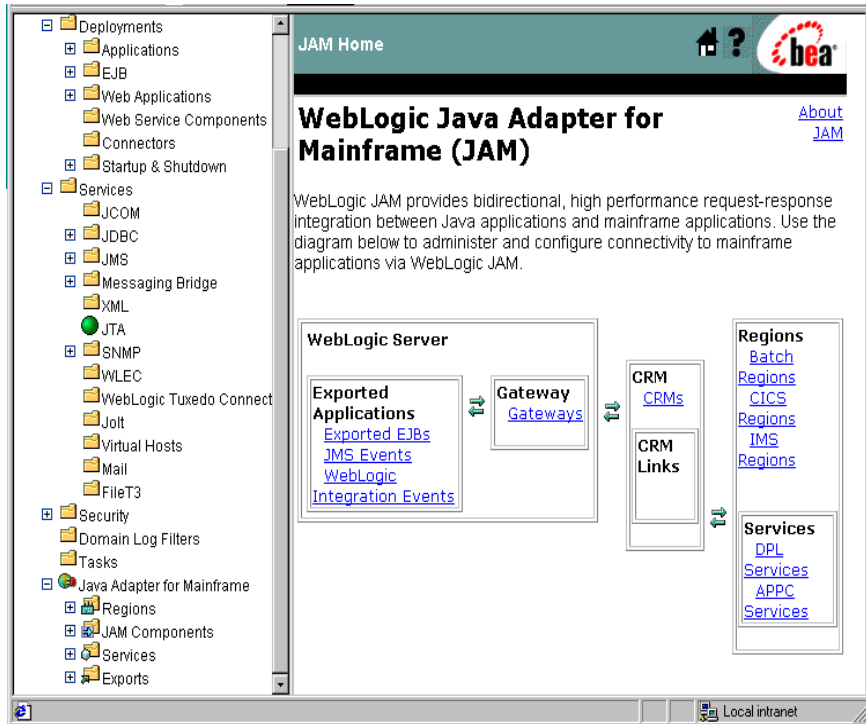
3. Change the JMS Event parameters to the appropriate values.

A check in the **Local Service Enabled** box makes the service available for use. Removing the check from the **Local Service Enabled** box disables the service.

4. Click **Apply** to save your changes.

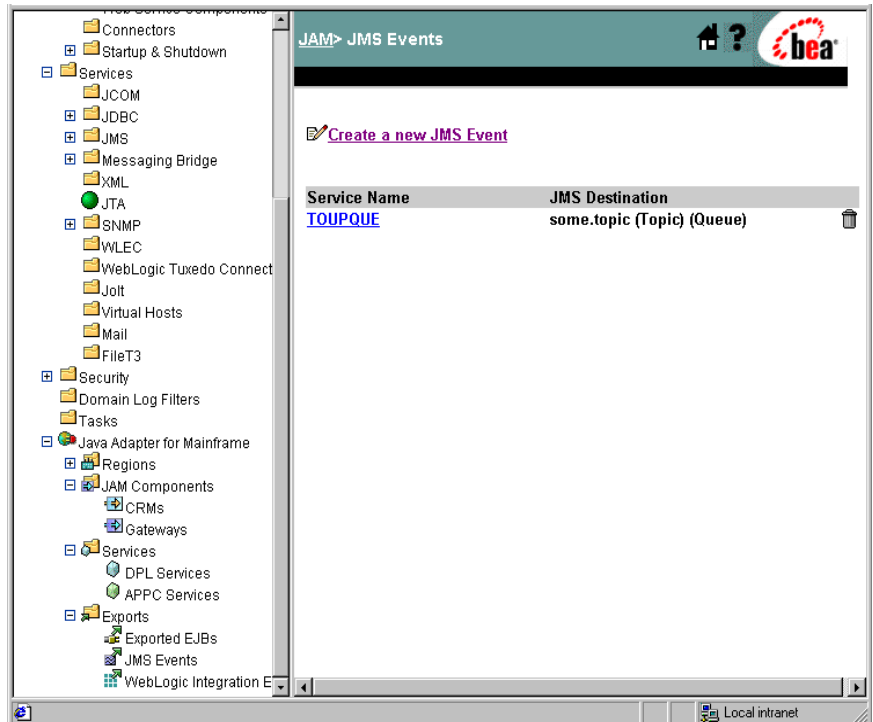
### Deleting a JMS Event

To delete a JMS Event, perform the following steps:

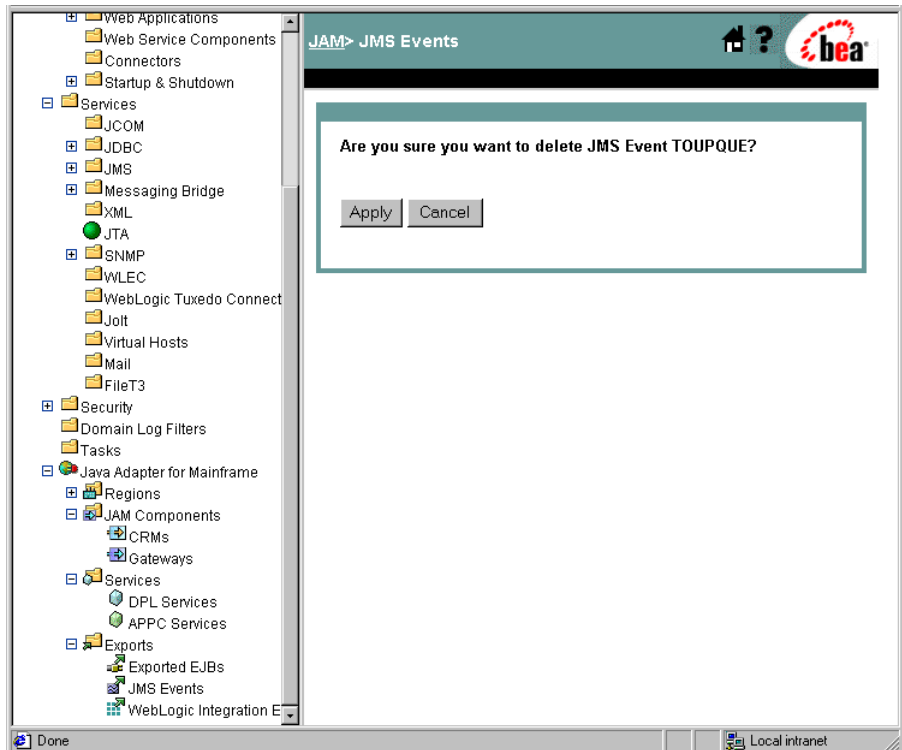


## 5 Modifying Your Configuration

1. From the WebLogic JAM home page, click **JMS Events**. The List JMS Events page displays.



2. Click the trash can icon.



3. Click **Apply** to finish deleting the JMS Event.

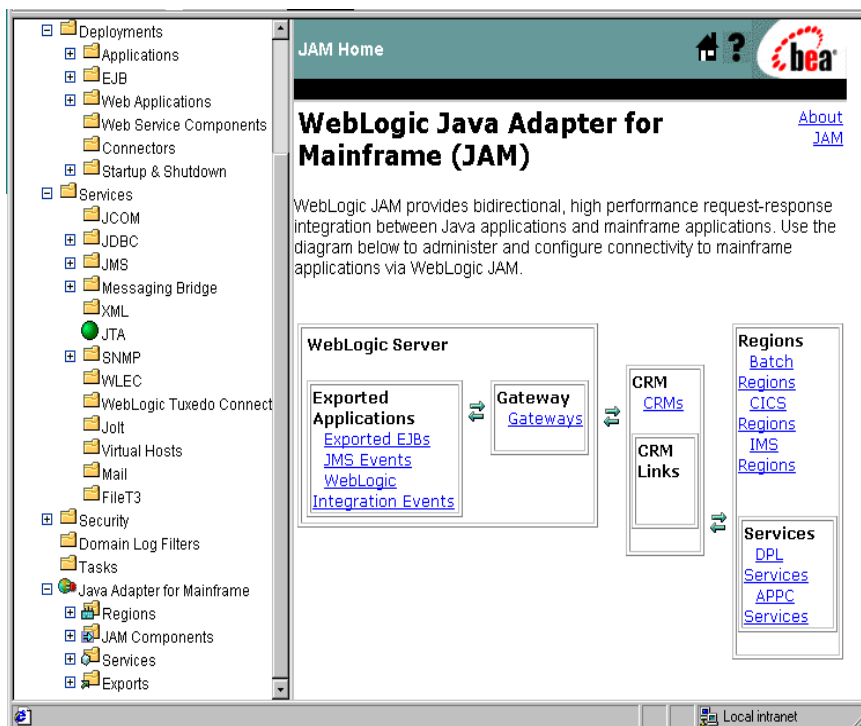
## Modifying a WebLogic Integration Event Definition

To modify a WebLogic Integration Event definition, you have the following options:

- [Editing a WebLogic Integration Event](#)
- [Deleting a WebLogic Integration Event](#)

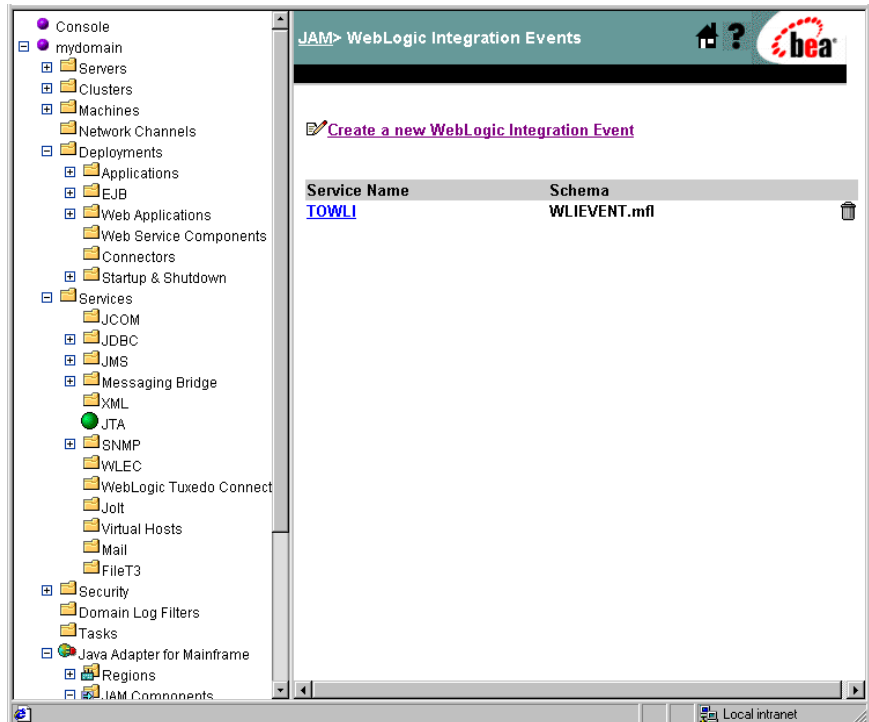
### Editing a WebLogic Integration Event

To edit a WebLogic Integration Event, perform the following steps:

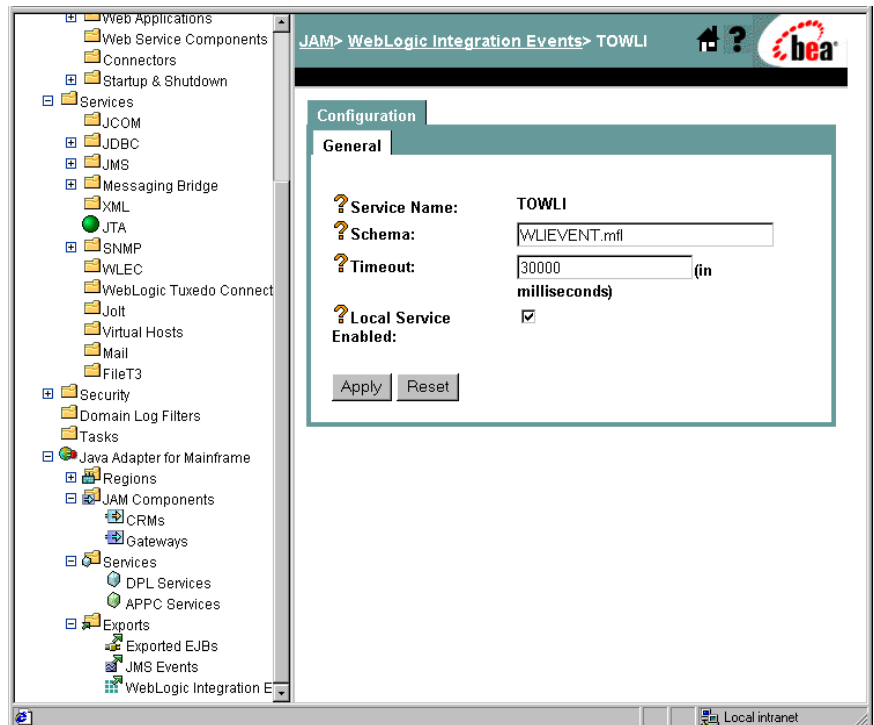




1. From the WebLogic JAM home page, click **WebLogic Integration Events**. The List WebLogic Integration Events page displays.



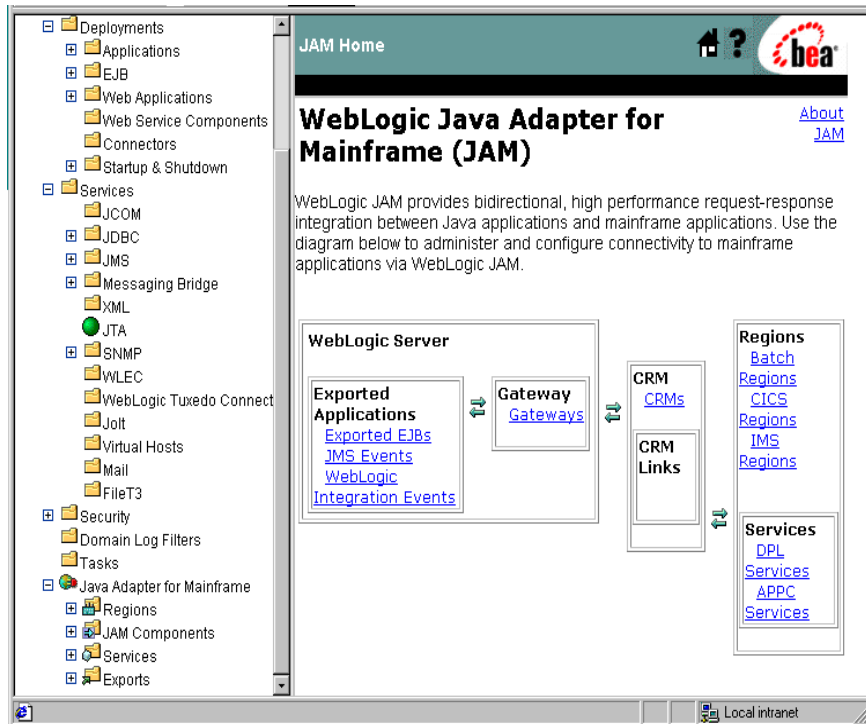
2. Click the WebLogic Integration Event that you want to edit.



3. Change the WebLogic Integration parameters to the appropriate values.  
A check in the **Local Service Enabled** box makes the service available for use. Removing the check from the **Local Service Enabled** box disables the service.
4. Click **Apply** to save your changes.

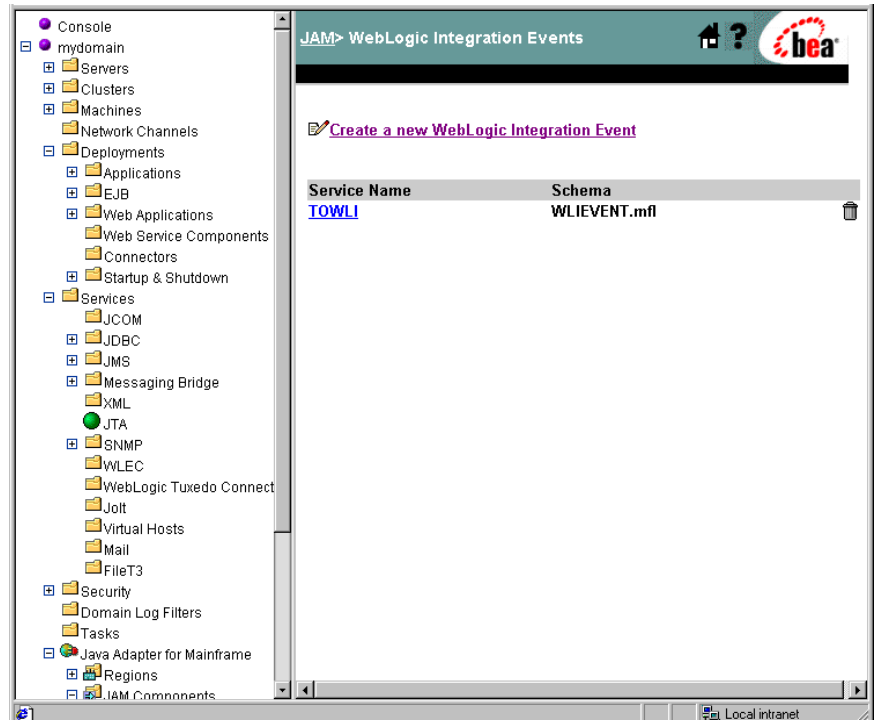
### Deleting a WebLogic Integration Event

To delete a WebLogic Integration Event, perform the following steps:

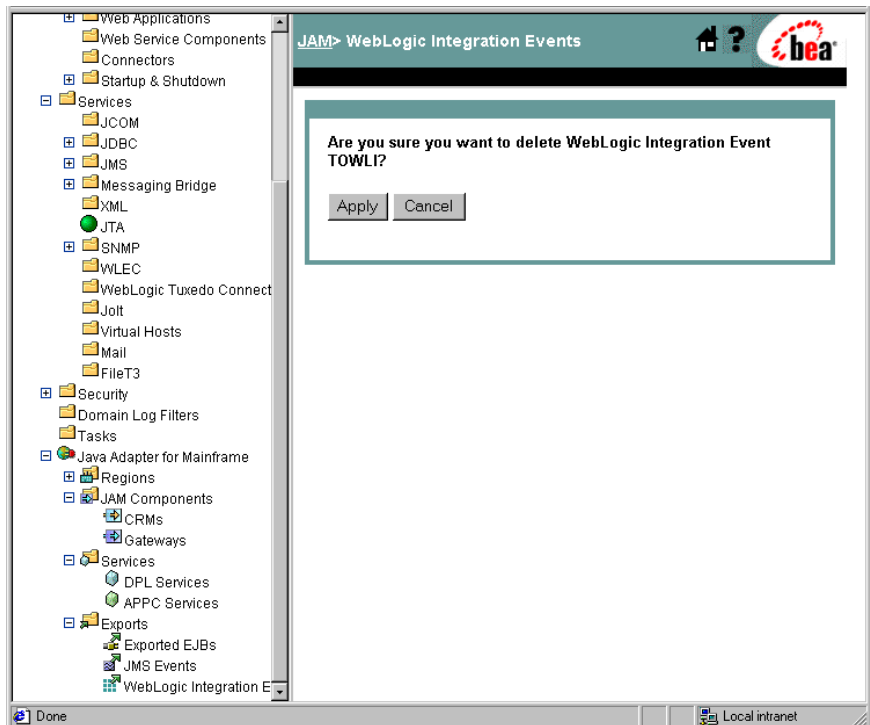


## 5 *Modifying Your Configuration*

1. From the WebLogic JAM home page, click **WebLogic Integration Events**. The List WebLogic Integration Events page displays.



2. Click the trash can icon.



3. Click **Apply** to finish deleting the WebLogic Integration Event.



# 6 Administration and Diagnostics Using the WebLogic Administration Console

The WebLogic Administration Console provides you with the ability to perform administrative and diagnostic tasks for the CRM, the Gateway, and various services. Using the WebLogic Administration Console, you can quickly and easily perform the following administrative tasks:

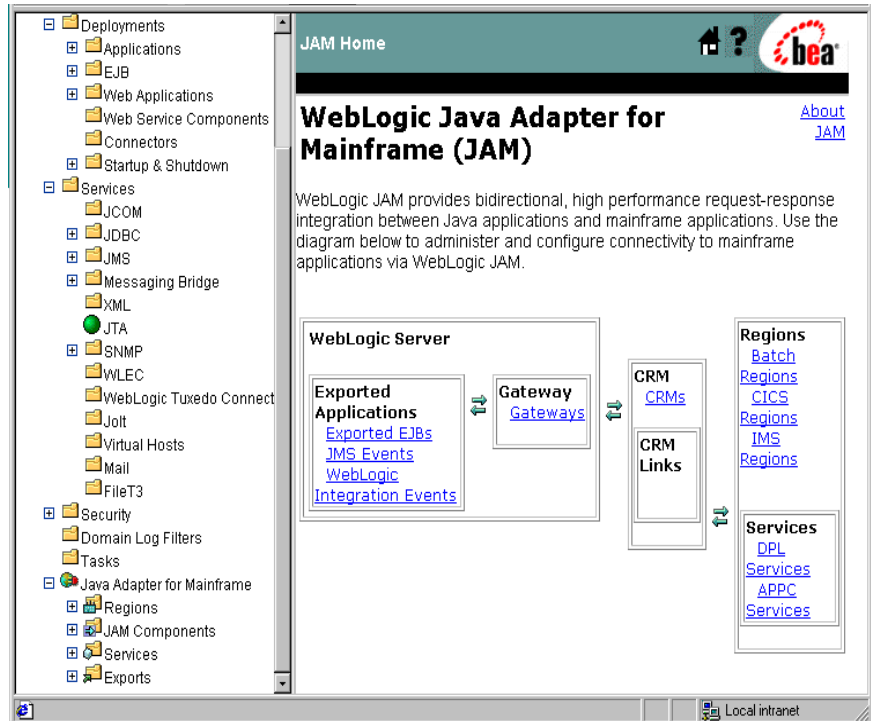
- [Monitoring WebLogic JAM CRM Status](#)
- [Modifying CRM Trace Level Settings](#)
- [Setting APPC API Tracing in the WebLogic Administration Console](#)
- [Monitoring CRM Links](#)
- [Starting and Stopping CRM Links](#)
- [Gateway Administration and Diagnostics](#)
- [Enabling a Service](#)

- [Enabling an Exported WebLogic Application](#)

# Monitoring WebLogic JAM CRM Status

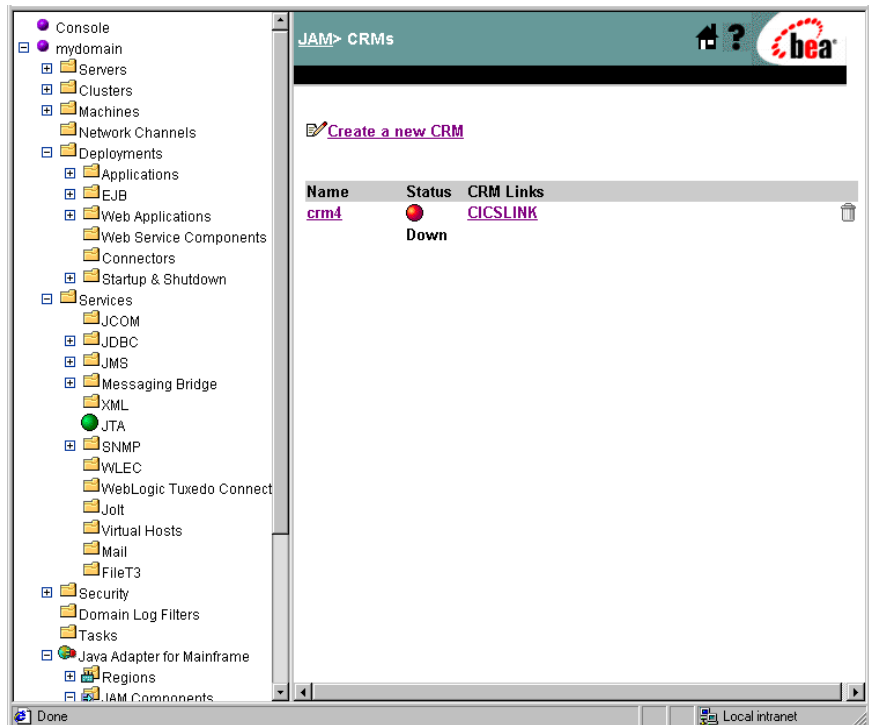
You can use the WebLogic Administration Console to monitor whether a CRM has been started or stopped. For information about starting and stopping the CRM, see [“CRM Administration.”](#)

To monitor WebLogic JAM CRM status from the WebLogic Administration Console:



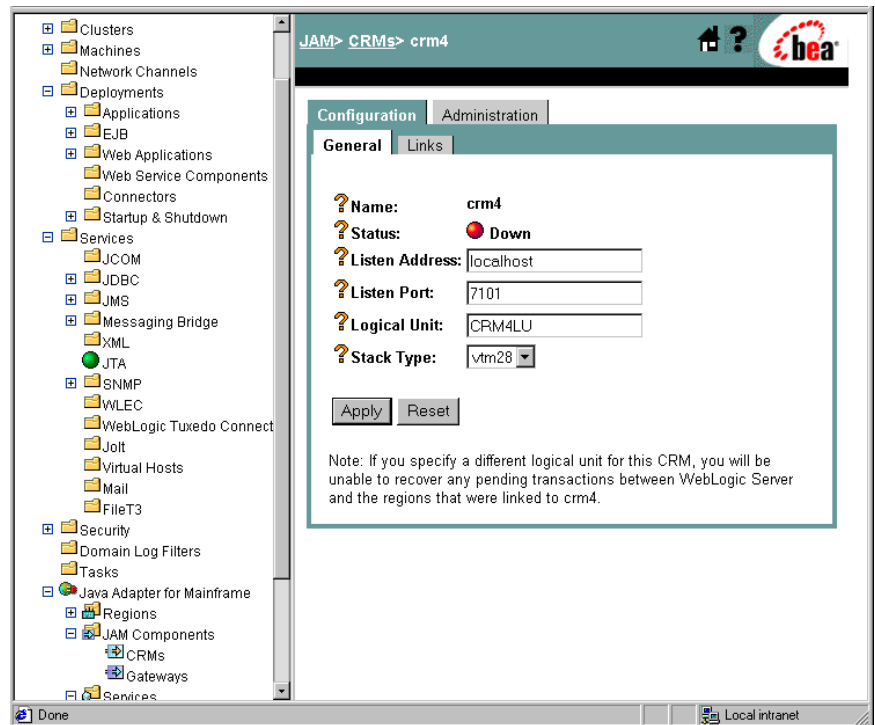


1. From the WebLogic JAM home page, click **CRM**. The List CRM page displays.

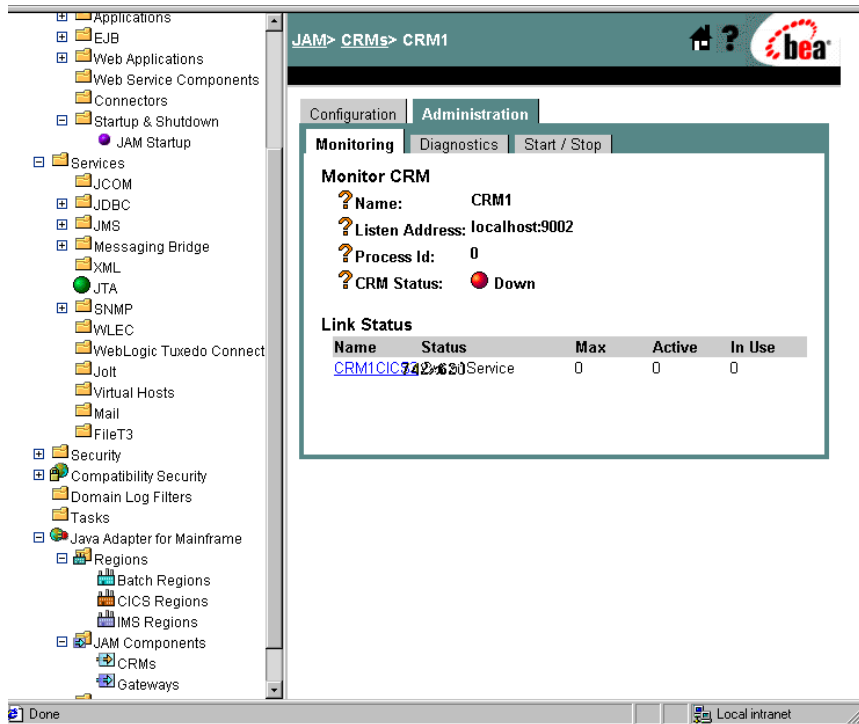


## 6 Administration and Diagnostics Using the WebLogic Administration Console

2. Click the CRM you want to monitor.



3. Click the Administration tab in the CRM links page. The Monitor a CRM page displays.



On the Monitoring CRM page, the following **Link Status** information may display beside the name of the CRM Link Name:

**Table 6-1 CRM Link Status**

Link Status	Description
Status	Whether the link is In Service or Out of Service. Also whether the link is: <ul style="list-style-type: none"> <li>Acquired</li> <li>Starting</li> <li>Released</li> <li>Free</li> </ul>
Max	The maximum number of sessions for the link.

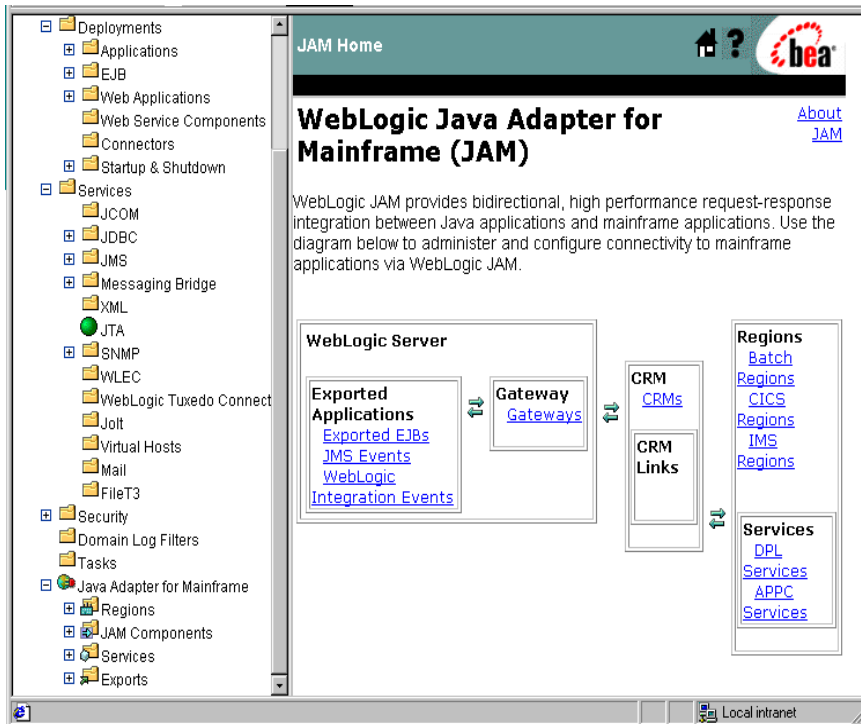
Link Status	Description
Active	The number of sessions that are currently active for the link.
In Use	The number of sessions that are currently in use on the link.

# Modifying CRM Trace Level Settings

The CRM can output a diagnostic trace file that describes its run-time activity. The trace level selected determines the amount of detail included in the tracing information.

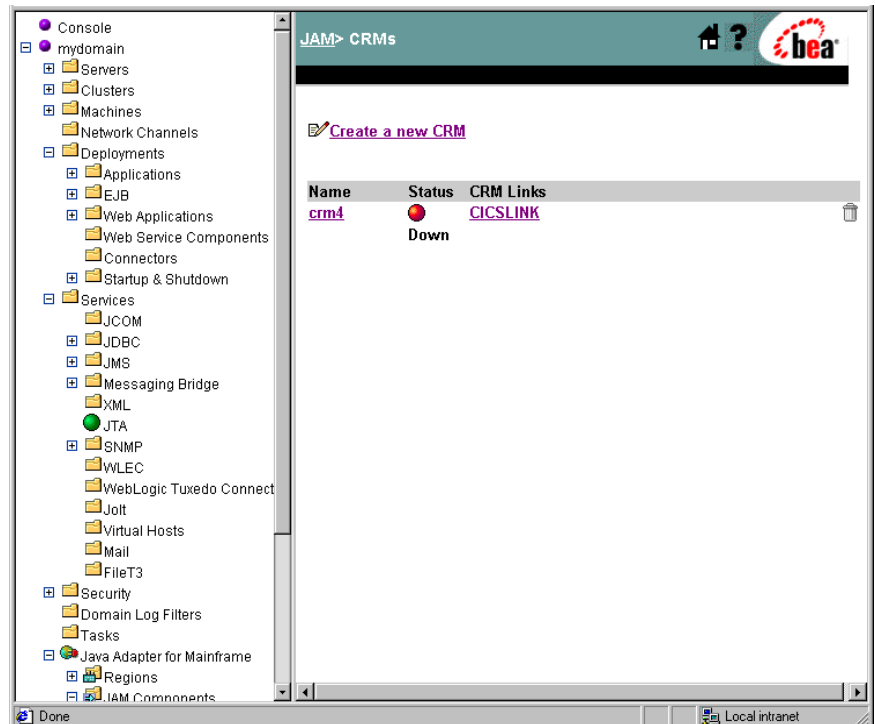
**Note:** The CRM tracing can be useful in diagnosing problems but should not be enabled in production systems.

To modify the trace level settings for the CRM:

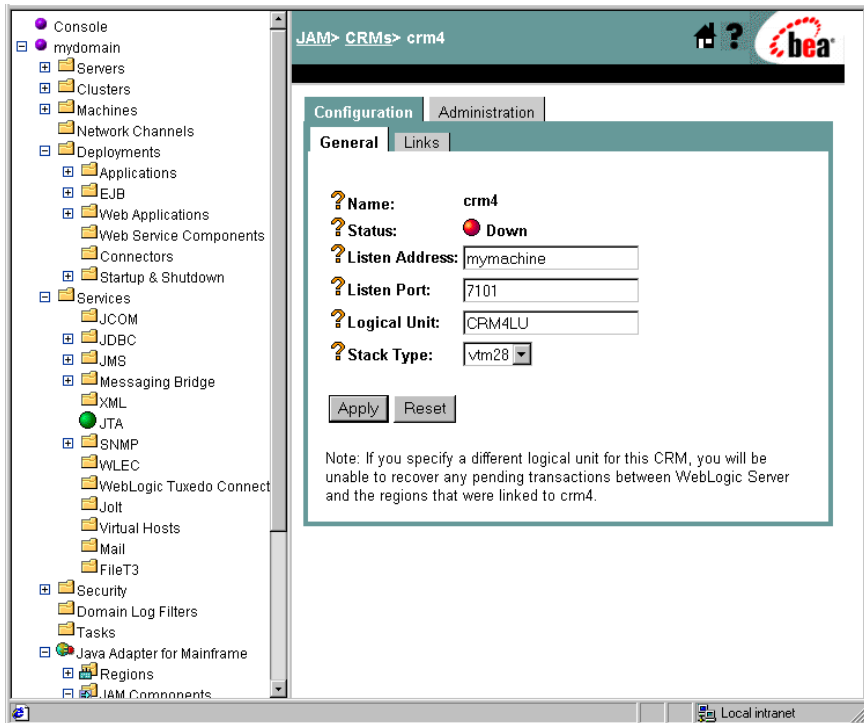


## 6 Administration and Diagnostics Using the WebLogic Administration Console

1. From the WebLogic JAM home page, click **CRM**. The List CRM page displays.

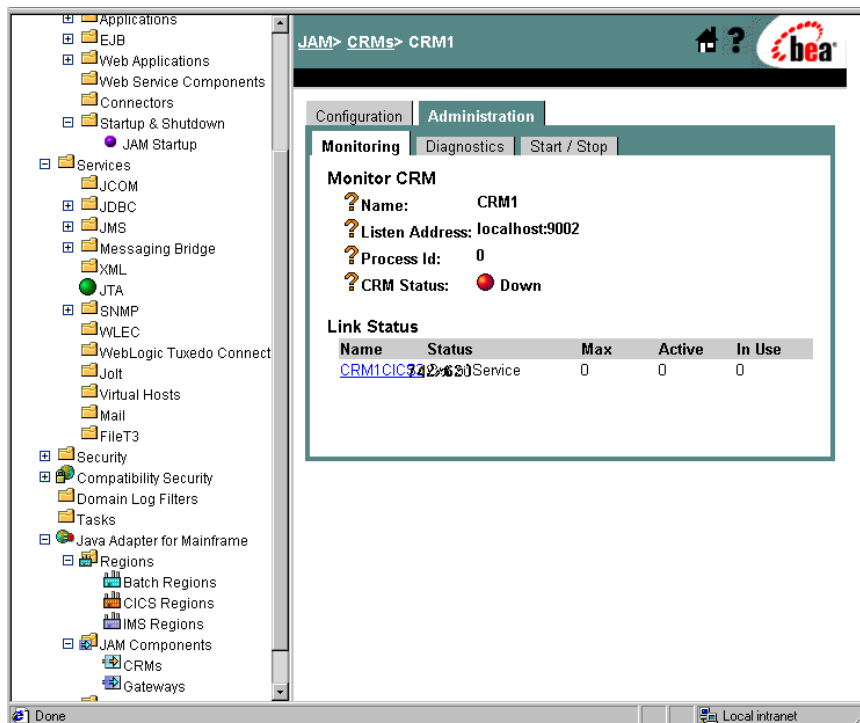


2. Click the CRM link you want to you want to modify.



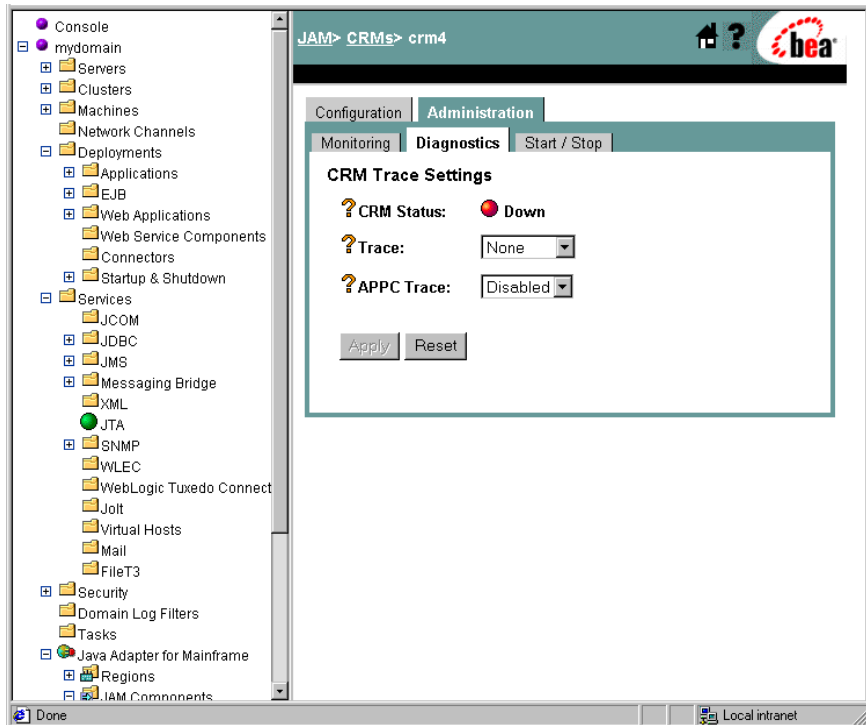
## 6 Administration and Diagnostics Using the WebLogic Administration Console

3. Click the Administration tab in the CRM links page. The Monitor a CRM page displays.





- Click the Diagnostics tab. The CRM Trace Settings page displays.



- Change the field values as appropriate. Click the question mark to the left of the field names for a description of the field. The following values are options from the **Trace** drop-down box.

**Table 6-2 CRM Trace Settings**

Trace Option Name	Description
None	No tracing.
Minimum	The CRM traces only major events and is sufficient only to determine the sequence of application conversations.
Medium	The CRM also traces all APPC verbs and WebLogic JAM/CRM communication.

## 6 Administration and Diagnostics Using the WebLogic Administration Console

---

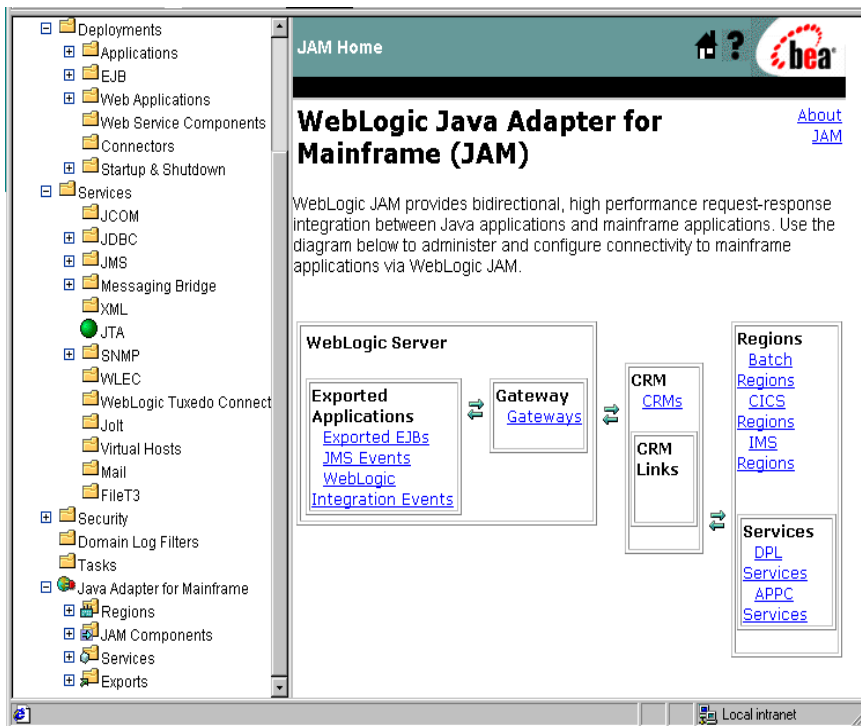
Trace Option Name	Description
Maximum	The CRM also traces all I/O buffers.

6. When you have finished editing the field values, click **Apply**.

**Note:** Clicking **Reset** causes the field values to be reset to their original settings.

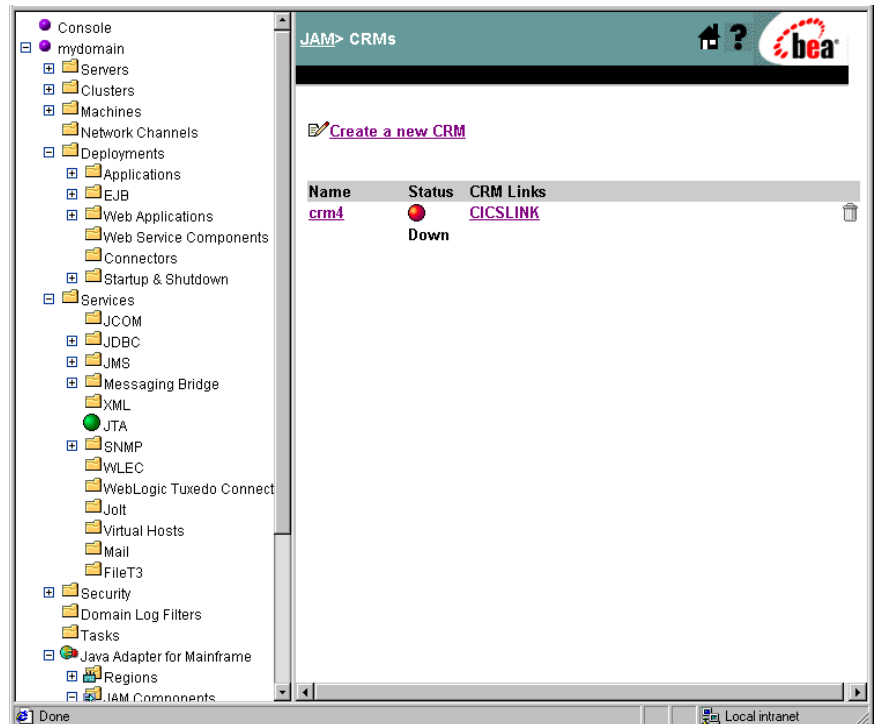
# Setting APPC API Tracing in the WebLogic Administration Console

To set the APPC API tracing from the WebLogic Administration Console:

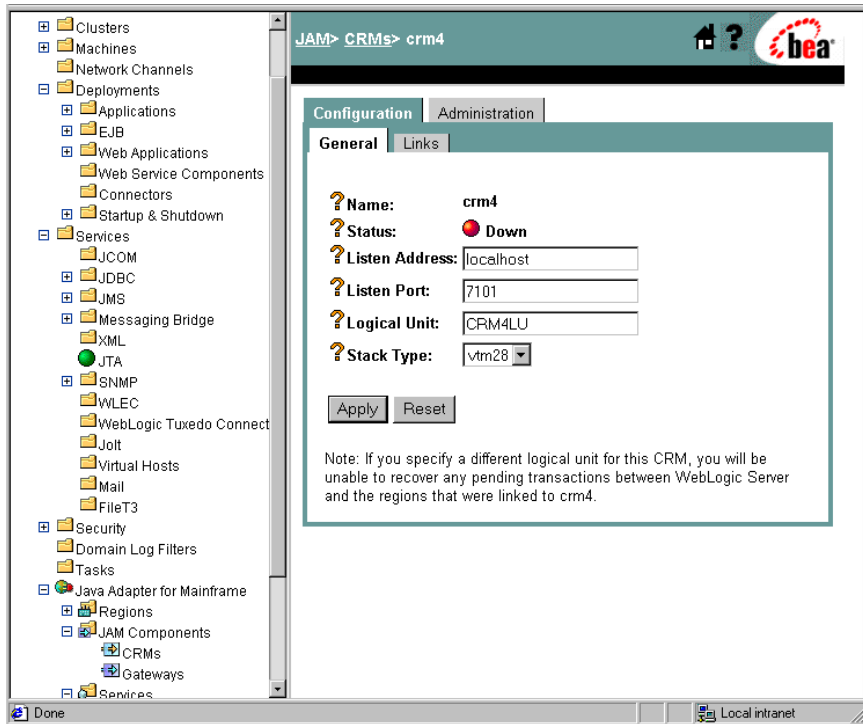


## 6 Administration and Diagnostics Using the WebLogic Administration Console

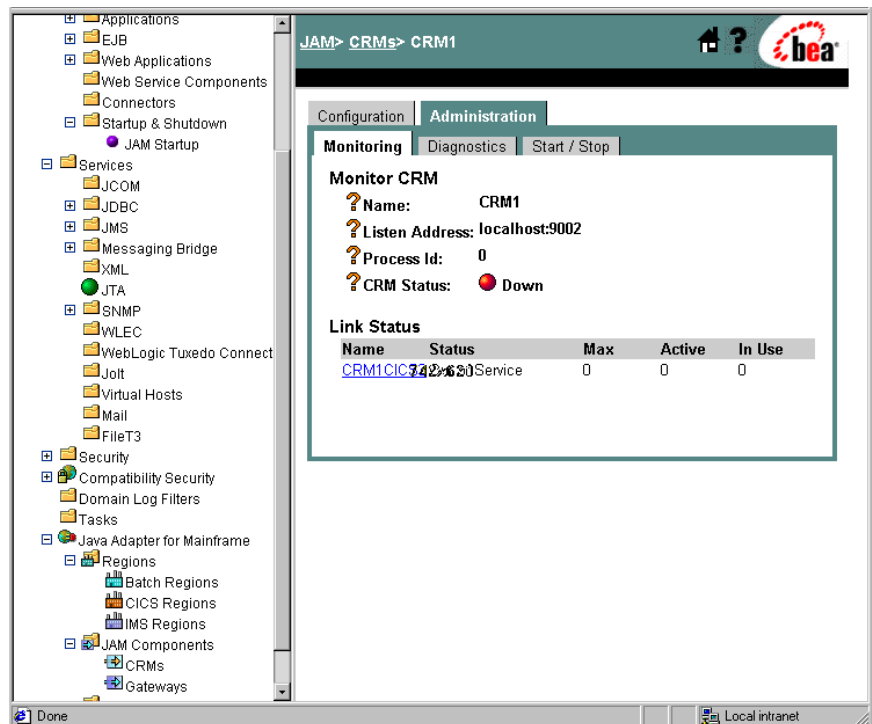
1. From the WebLogic JAM home page, click **CRM**. The List CRM page displays.



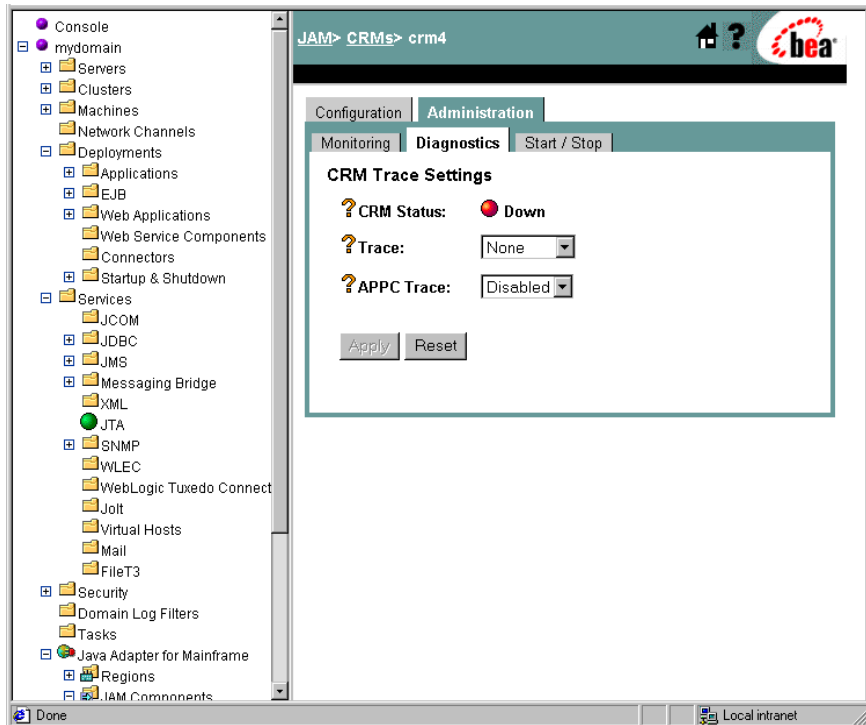
2. Click the appropriate CRM.



3. Click the Administration tab.



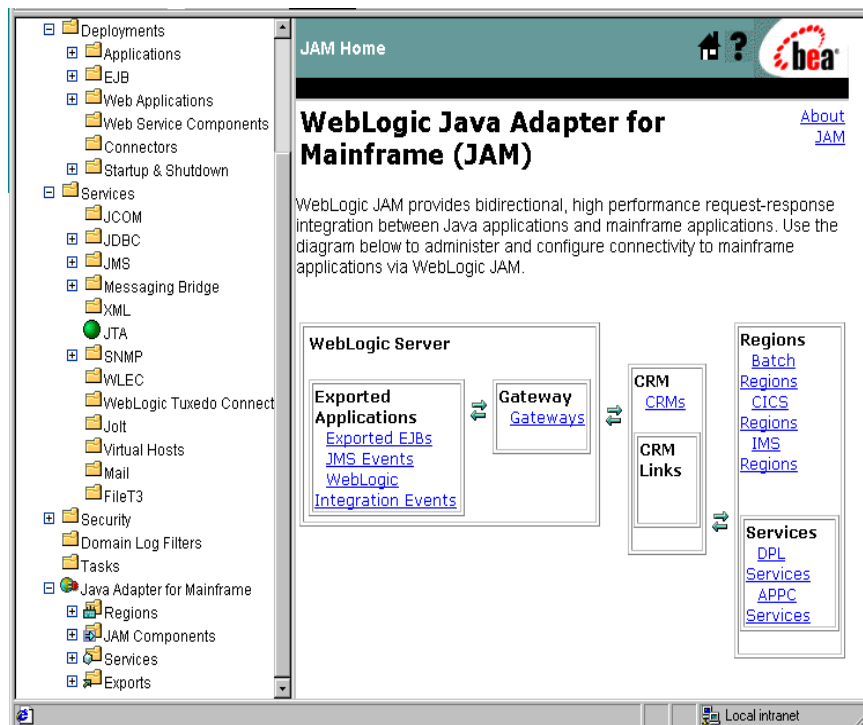
- Click the Diagnostics tab.



- Use the drop-down menu to select either **Enabled** or **Disabled** in the **APPC Trace** field.
- Click **Apply**.

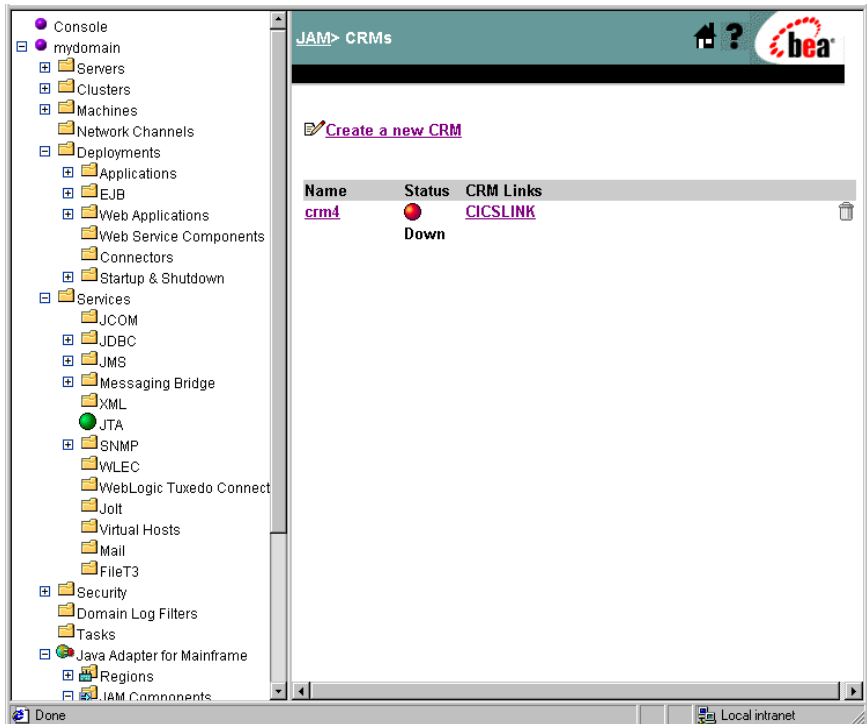
## Monitoring CRM Links

To monitor the status for a CRM Link:



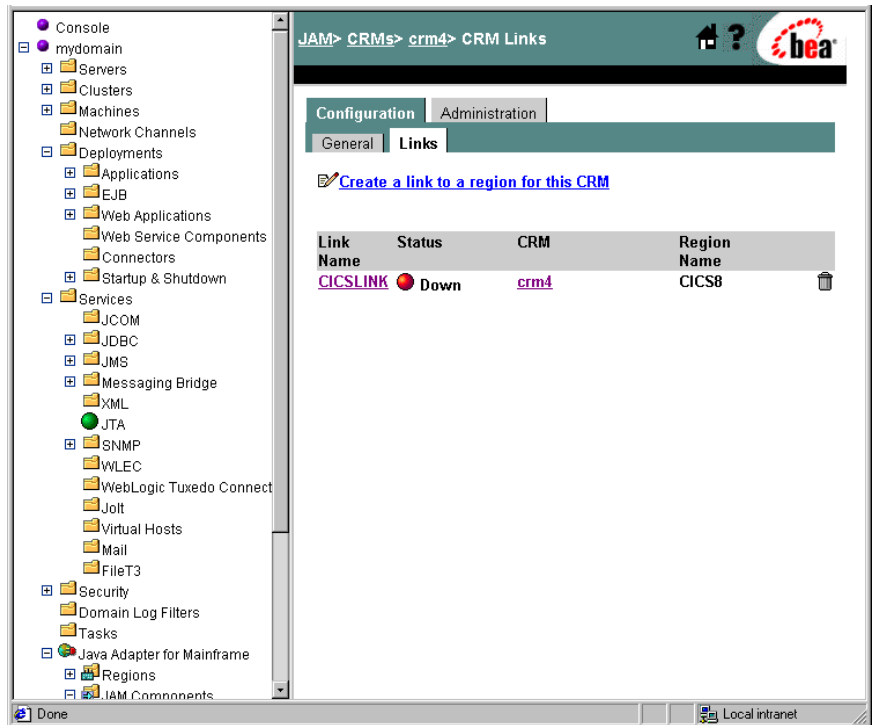
1. From the WebLogic JAM home page, click **CRM**. The List CRM page displays.



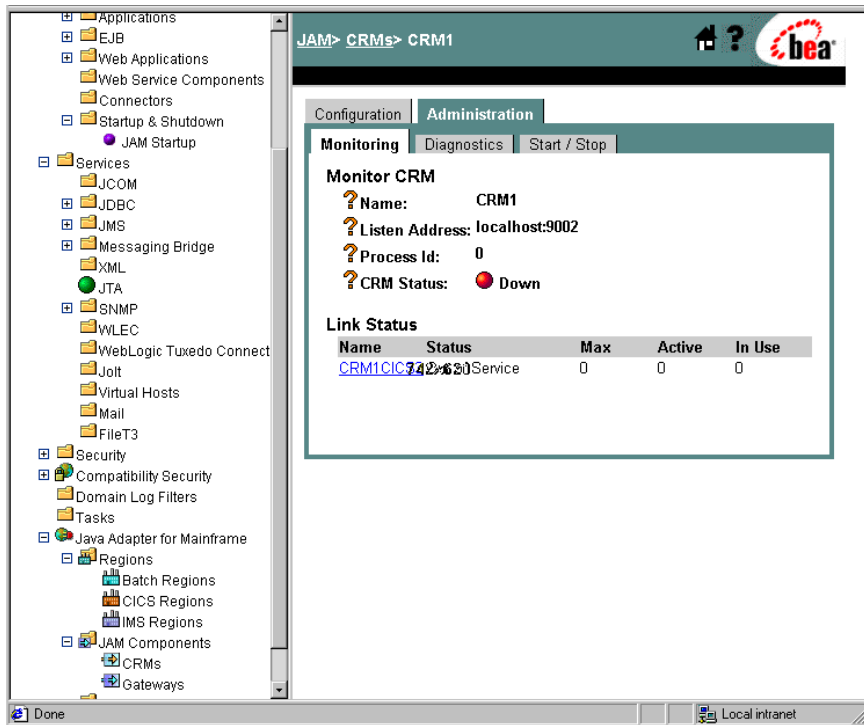


## 6 Administration and Diagnostics Using the WebLogic Administration Console

2. Click the CRM that has the CRM Link you want to monitor.



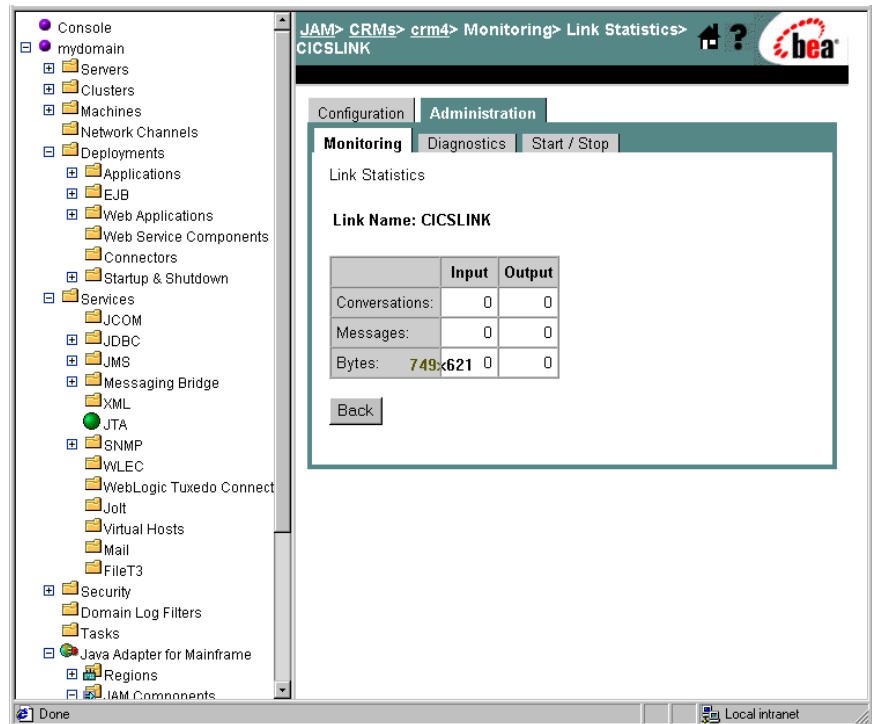
3. Click the Administration tab in the CRM page. The Monitor a CRM page displays.



- Click the name of the CRM Link you want to monitor. The Monitor Link Statistics page displays with the following information:

**Table 6-3 CRM Link Statistics**

Statistics	Description
Input	Information received from the CRM.
Output	Information sent to the mainframe.
Conversations	The number of conversations completed.
Messages	The number of messages transferred.
Bytes	The number of bytes transferred.

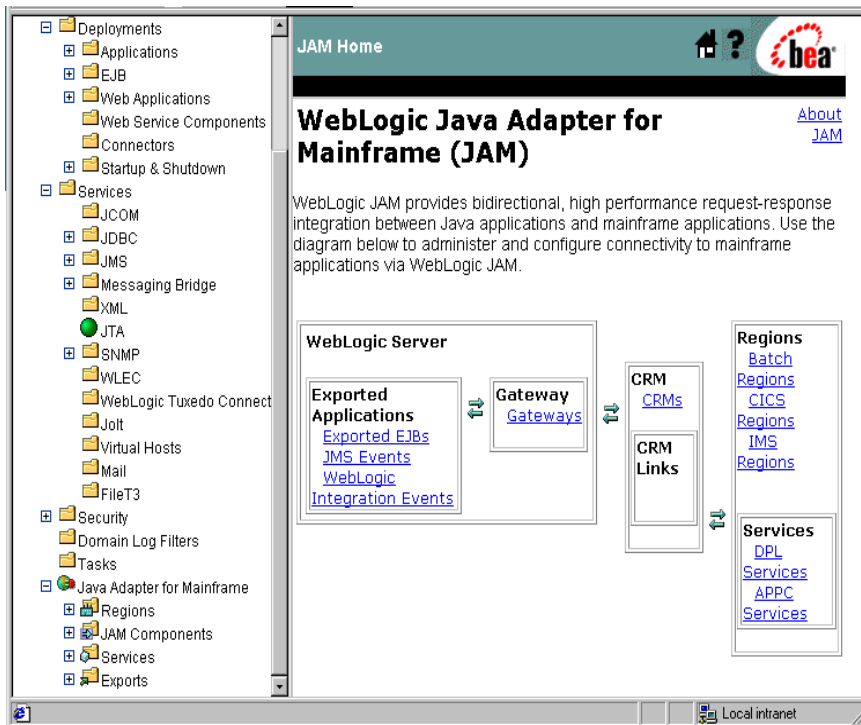


5. When you have finished monitoring the CRM Link, click **Back** to return to the Monitor a CRM page.

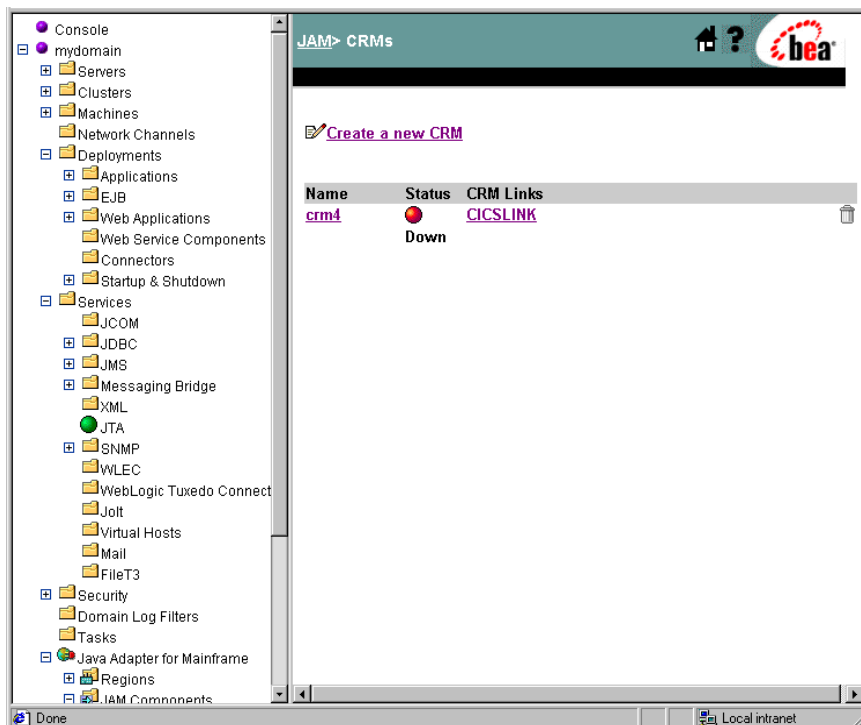
**Note:** The Monitor CRM Link Statistics page refreshes every second.

# Starting and Stopping CRM Links

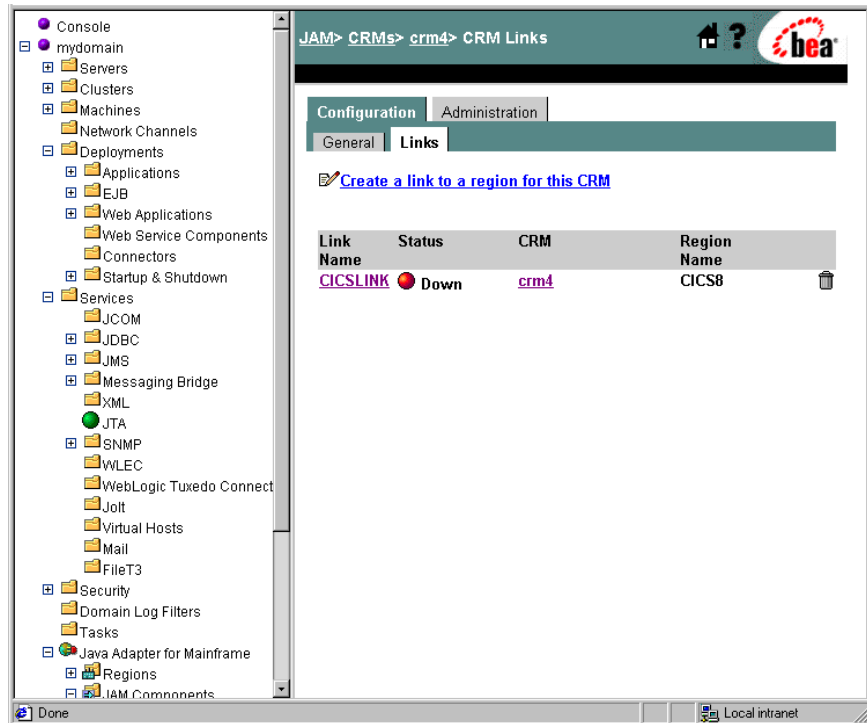
To start or stop a CRM link:



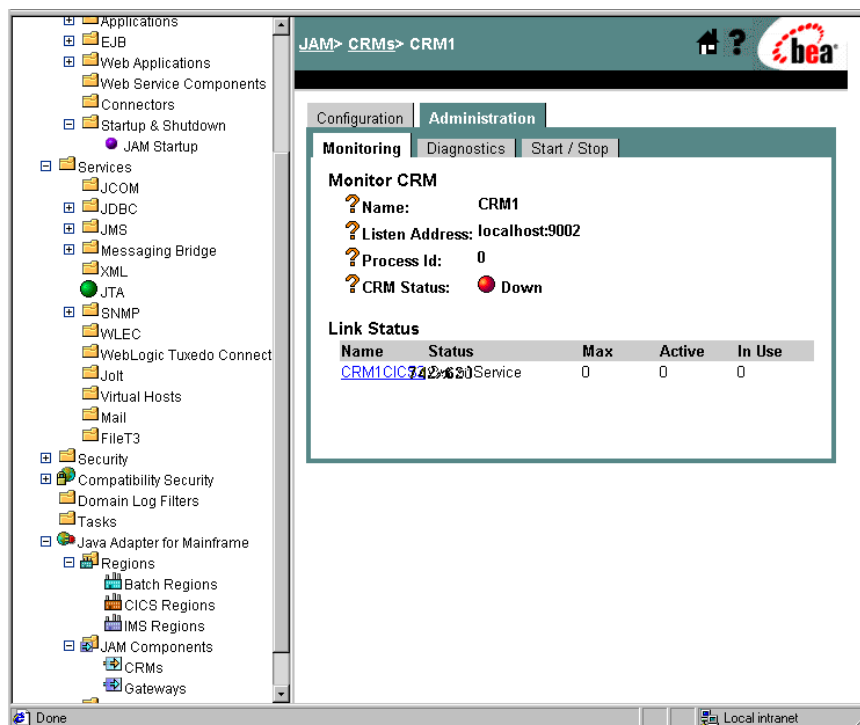
1. From the WebLogic JAM home page, click **CRMs**. The List CRM page displays.



2. Click the CRM Link you want to start or stop.

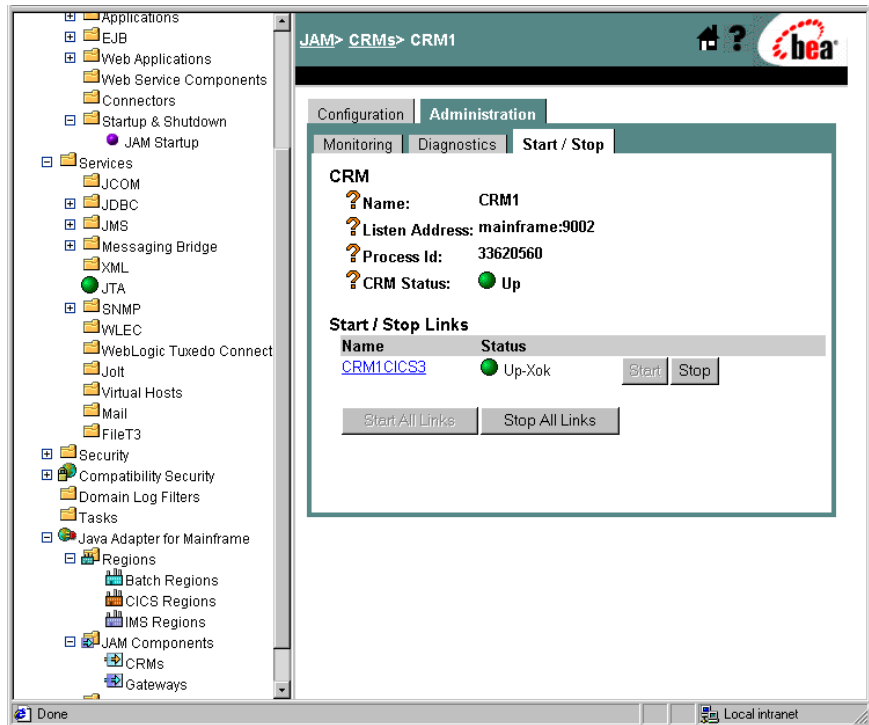


3. Click the Administration tab in the CRM page. The Monitor a CRM page displays.





4. Click the Start/Stop tab in the CRM Link Administration page. The link names associated with that CRM and the status of the links will display. The Status column will define the link status as Up or Down. If the link status is Up, it will be followed by a qualifier Xok or Xno.
  - Xok means that the link has transactions enabled.
  - Xno means that transactions are not enabled for the link.



5. Click **Start** next to the link name and status to start an individual CRM Link. The status of the link changes to UP.

or

Click **Stop** next to the link name and status to stop an individual CRM Link. The status of the link changes to Down.

6. Click Start All Links to start all links defined for this CRM.

or

Click Stop All Links to stop all links defined for this CRM.

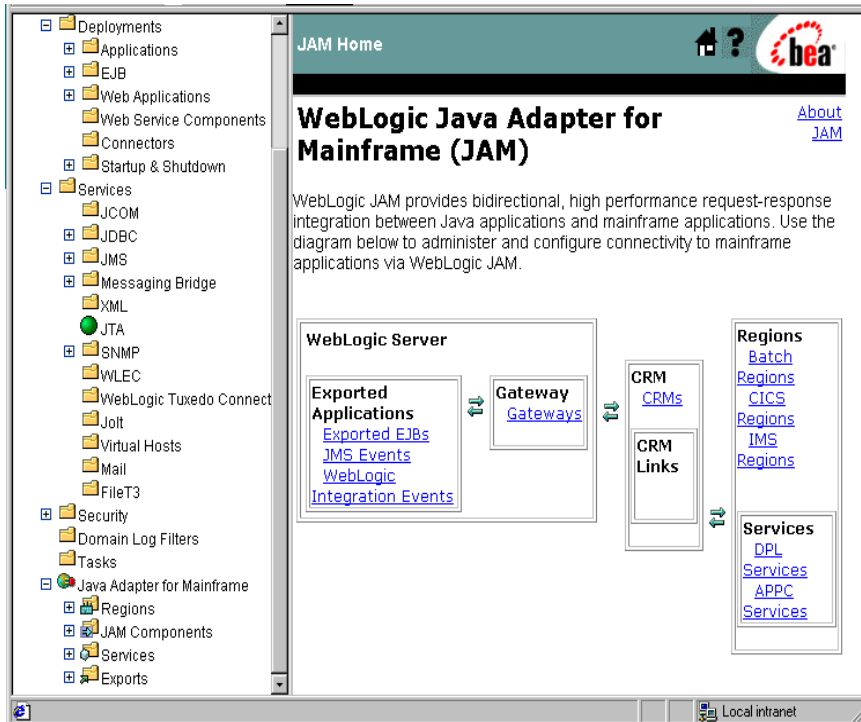
# Gateway Administration and Diagnostics

Use the WebLogic Administration Console to perform the following administration and diagnostic tasks for the Gateway:

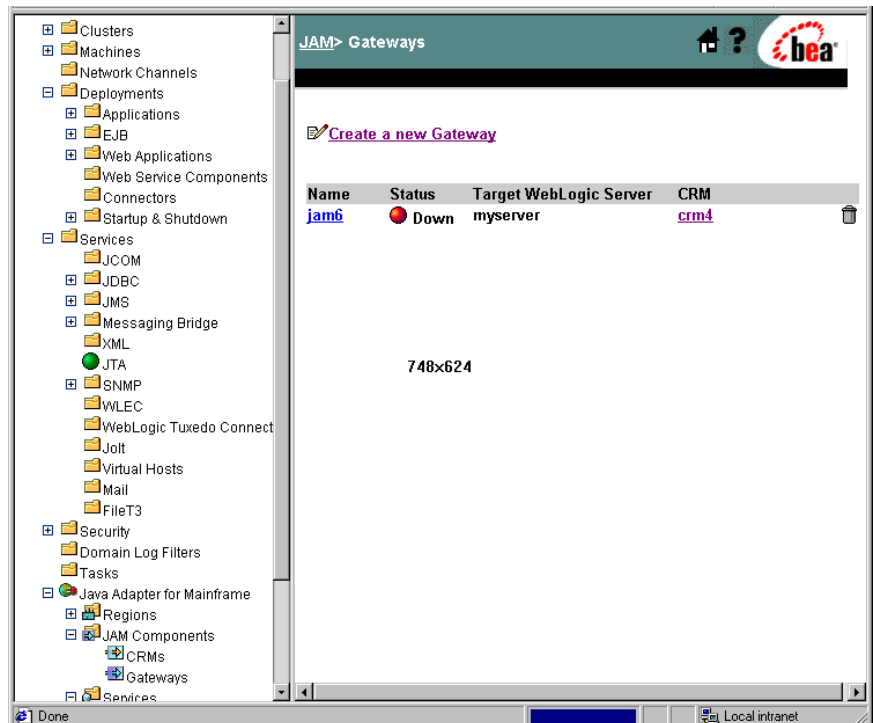
- [Listing Gateways](#)
- [Monitoring a Gateway](#)
- [Modifying Trace Level Settings for the Gateway](#)
- [Starting and Stopping a Gateway](#)

## Listing Gateways

To list a Gateway:

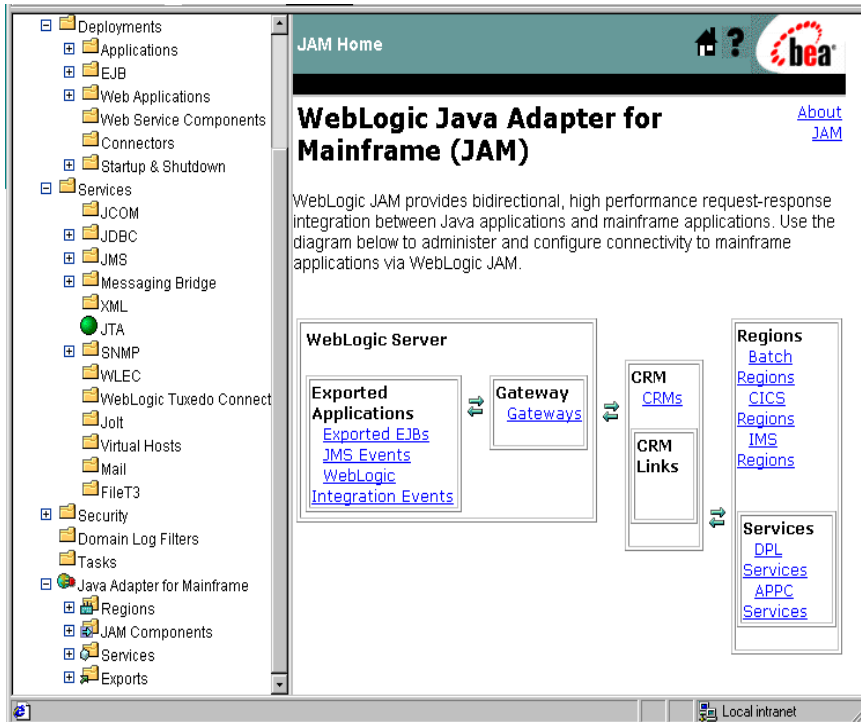


1. From the WebLogic JAM home page, click **Gateways**. The List Gateways page displays.

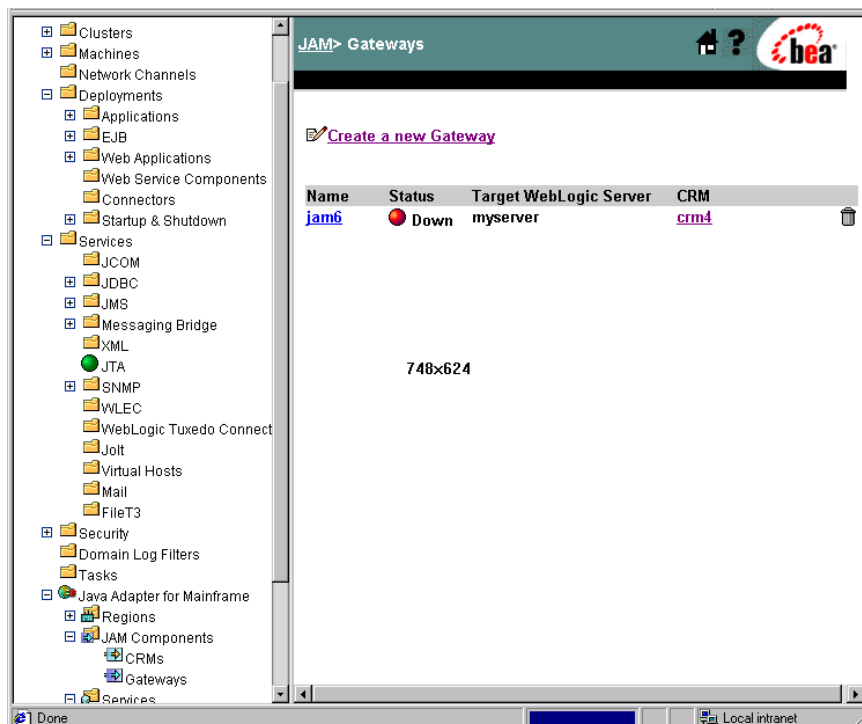


# Monitoring a Gateway

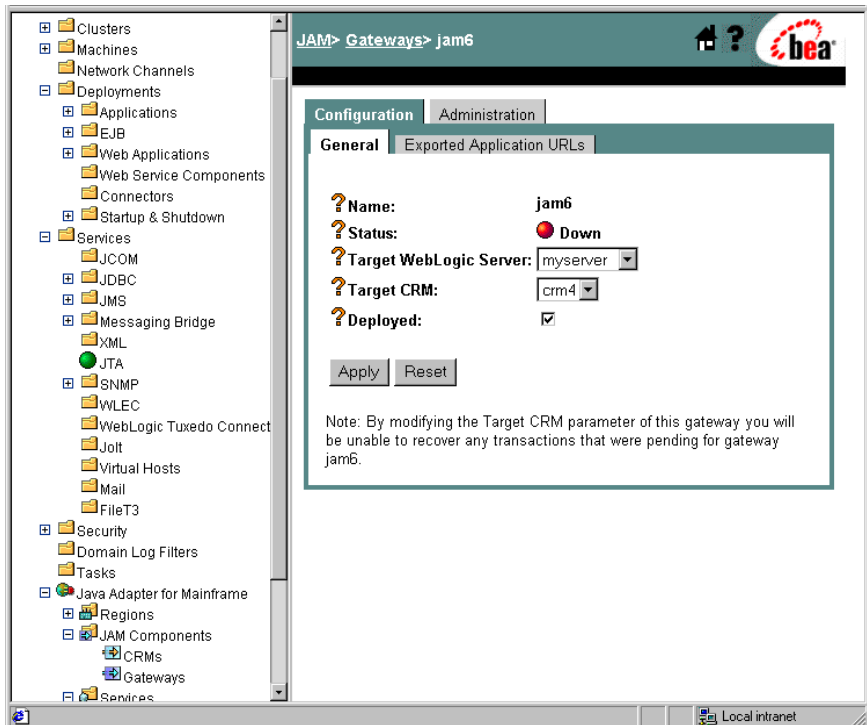
To monitor a Gateway:



1. From the WebLogic JAM home page, click **Gateways**. The List Gateways page displays.



2. Click the appropriate existing Gateway link you want to monitor.



- Click the Administration tab. The Monitor Gateway page displays. This page displays the name and current status of the Gateway. When the Gateway is up, statistics are displayed indicating how many requests have been successfully processed and the average time it took to process each request.

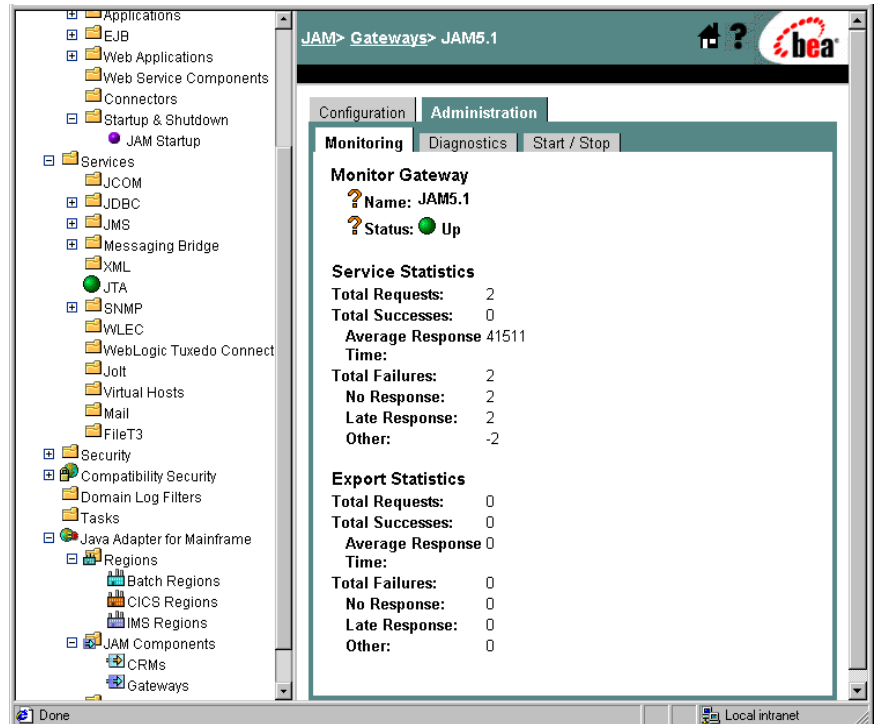


Table 6-4 provides a description of the statistics you can monitor for the WebLogic JAM Gateway.

**Table 6-4 Statistics for the WebLogic JAM Gateway**

WebLogic JAM Gateway Statistic	Description
Total Requests	The number of requests that have reached the gateway. This may be larger than the sum of successes and failures if some requests are still being processed.

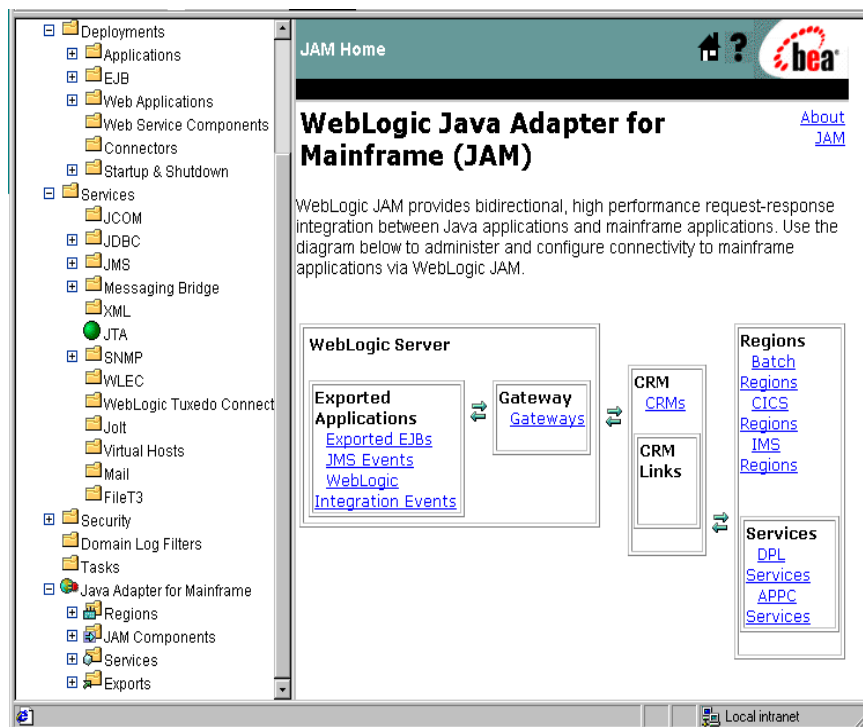


<b>WebLogic JAM Gateway Statistic</b>	<b>Description</b>
Total Successes	The number of requests that have successfully been processed to completion by the gateway. Application level failures may be reported as gateway successes.
Average Response Time	The average response time for all successful requests and some failures. Failures that fail before they are transmitted over the network do not affect this statistic. Timeouts do not affect this statistic until a late reply is received.
Total Failures	The total number of failures of any kind.
No Response	The number of requests that have timed out and have never received a response of any kind.
Late Response	The number of requests that timed out and then received a response.
Other	The number of request that failed other than by timeout.

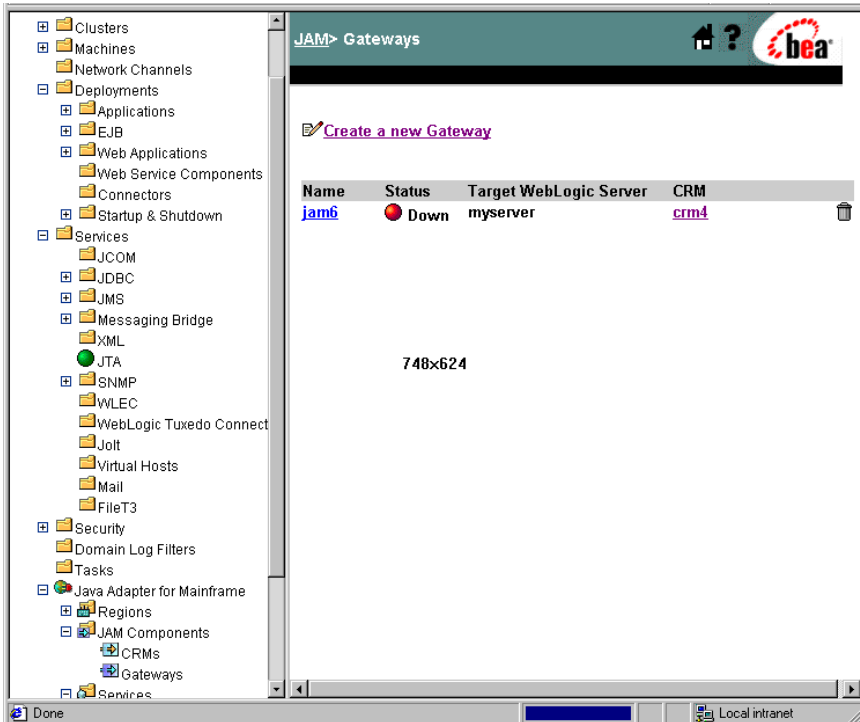
## Modifying Trace Level Settings for the Gateway

The Gateway diagnostics page allows you to set different kinds of tracing in the Gateway. When set, the tracing causes the WebLogic JAM Gateway to output diagnostic debugging information in the WebLogic Server log file. This information can be useful to diagnose problems, but it should not be enabled in production systems.

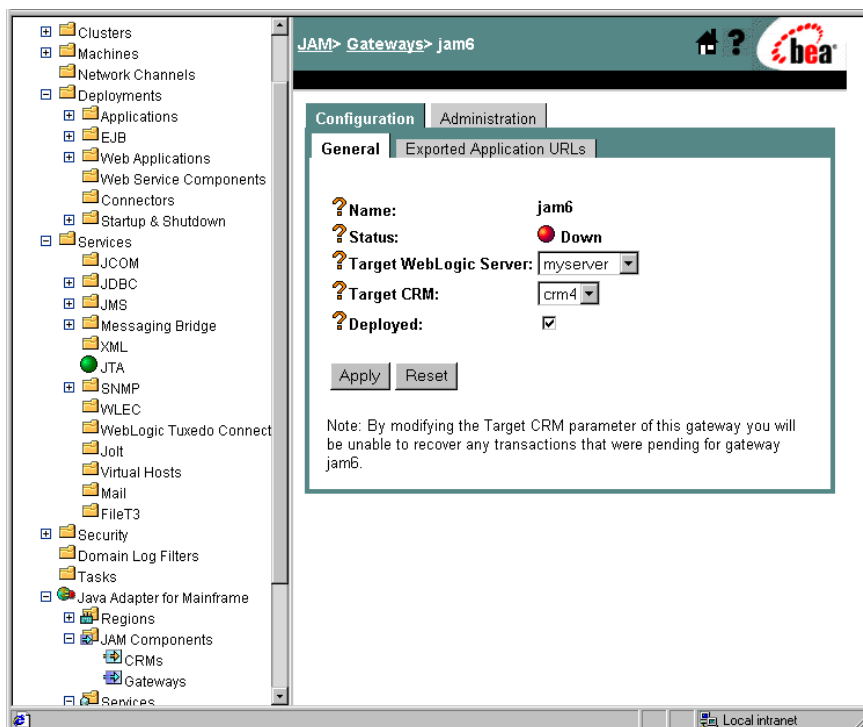
To modify the trace level settings for the Gateway:



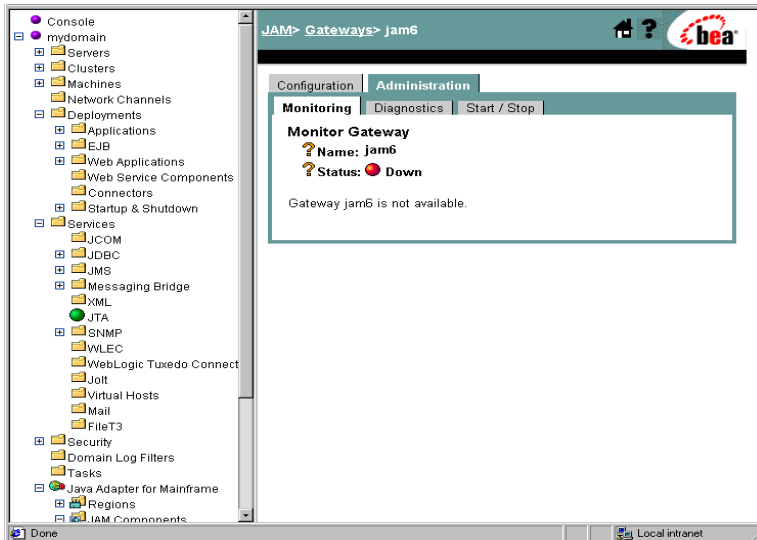
1. From the WebLogic JAM home page, click **Gateways**. The List Gateways page displays.



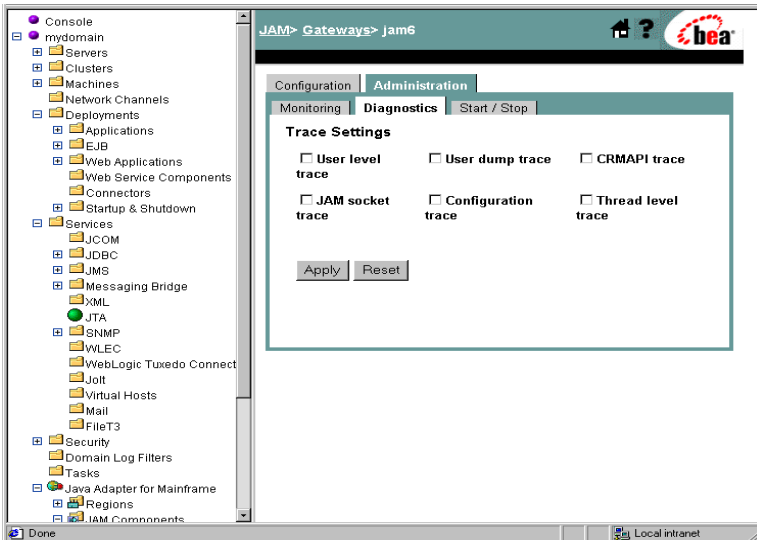
2. Click the link to the Gateway for which you want to monitor trace level settings.



- Click the Administration tab. The Monitor Gateway page displays.



- Click the Diagnostics tab. The Gateway Trace Settings page displays.



- Click next to the option to select the appropriate options.

**Table 6-5 Gateway Trace Options**

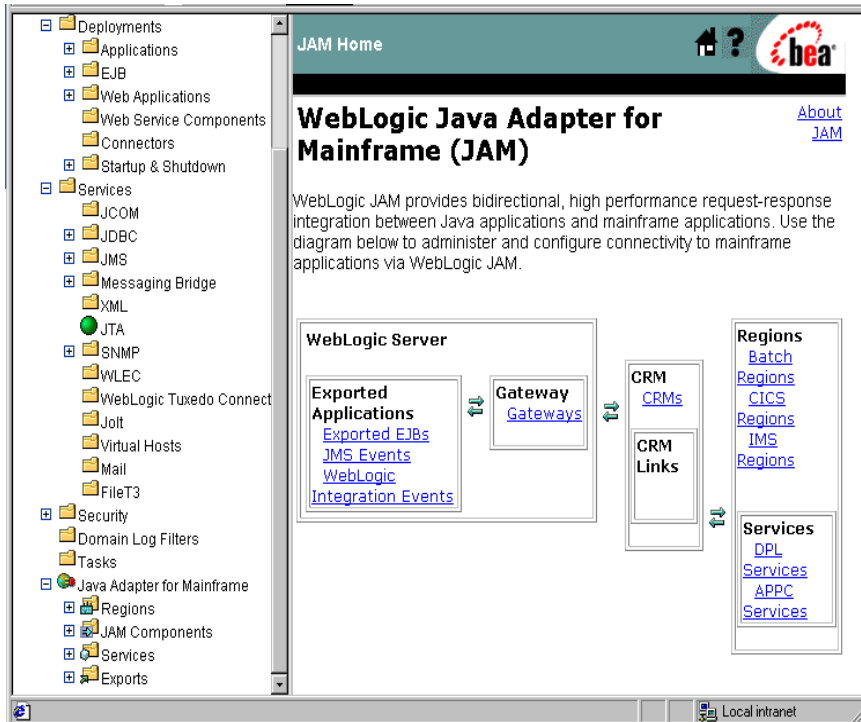
Trace Option Name	Description
User level trace	Produces trace records for the beginning and completion of all user requests, both to and from the mainframe. The completion message will indicate the success or failure of the request.
JAM socket trace	Produces trace records showing a hexadecimal dump of the data exchanged between the JAM gateway and the CRM.
User dump trace	Produces trace records with a hexadecimal dump of the user data associated with all user requests and replies. This trace level will also cause the trace records for User level trace to be produced.
Configuration trace	Produces trace records showing operations within the JAM Console and interactions between it and the JAM Gateway.
CRMAPI trace	Produces trace records showing the messages exchanged between the JAM gateway and the CRM.
Thread level trace	Produces trace records showing operations within the JAM Gateway related to its internal threads and subtasks.

6. When you have finished selecting the field options, click **Apply**.

**Note:** Clicking **Reset** causes the field values to be reset to their original settings.

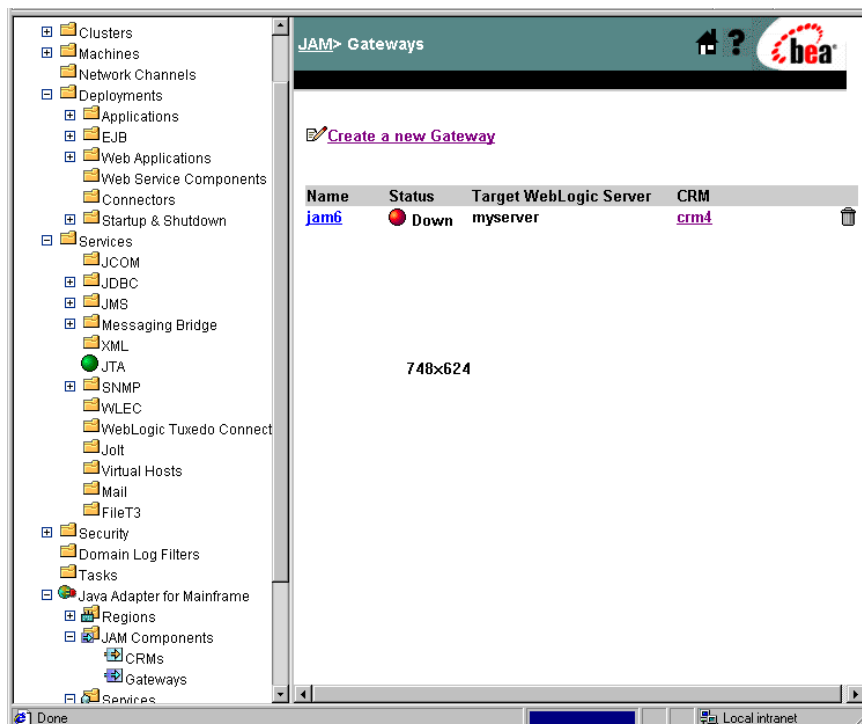
# Starting and Stopping a Gateway

To start or stop a Gateway:



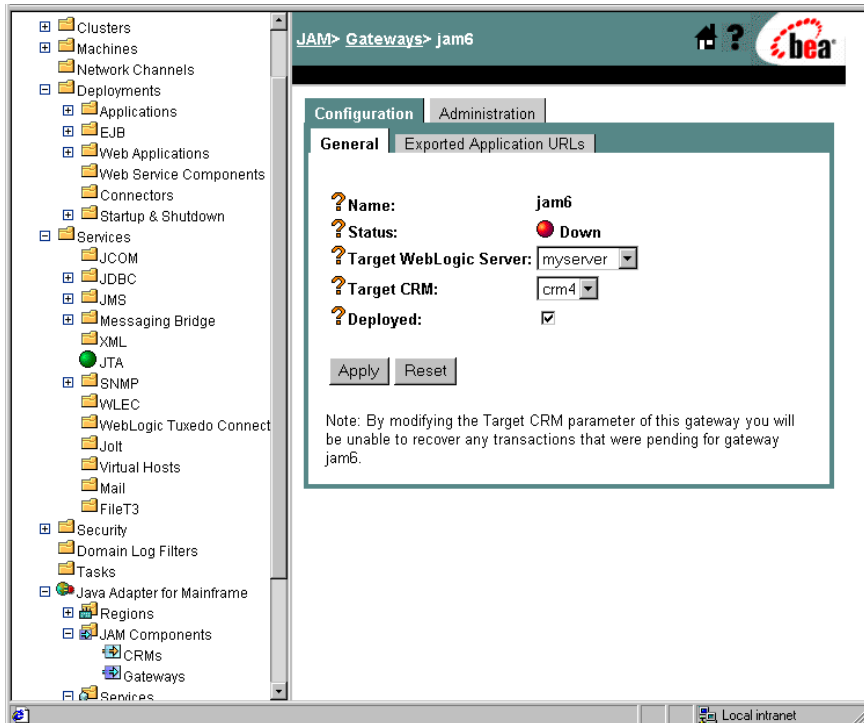
## 6 Administration and Diagnostics Using the WebLogic Administration Console

1. From the WebLogic JAM home page, click **Gateways**. The List Gateways page displays.

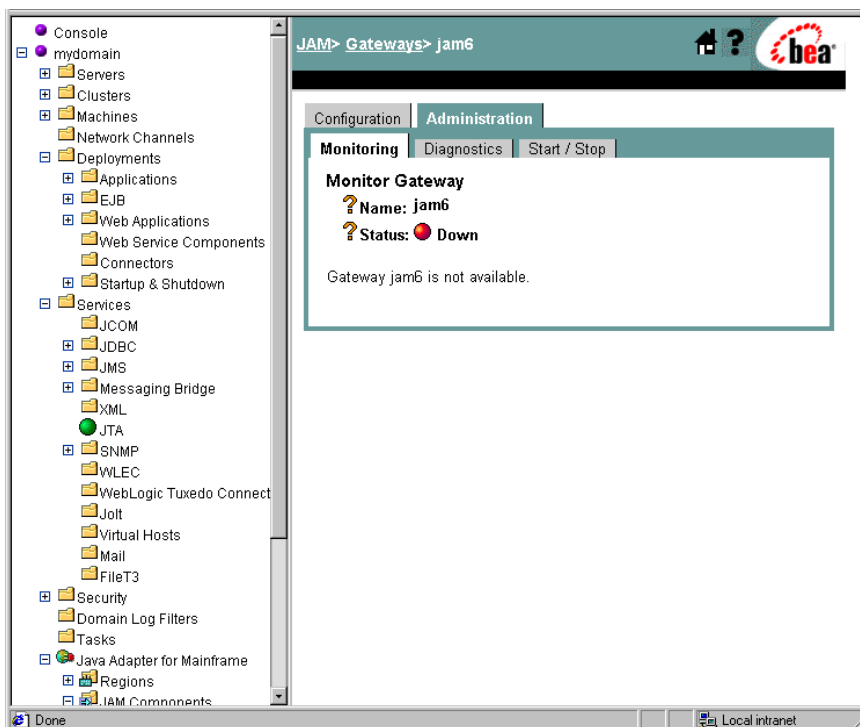




2. Click the appropriate Gateway link.



3. Click the Administration tab. The Monitor Gateway page displays.



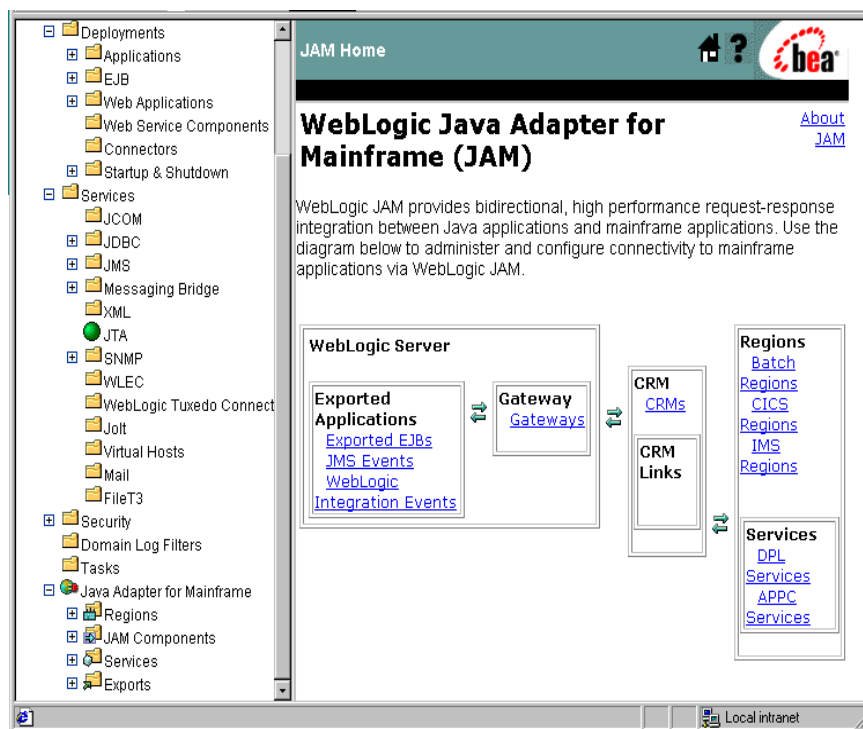
- Click the Start/Stop tab in the Gateway Administration page.



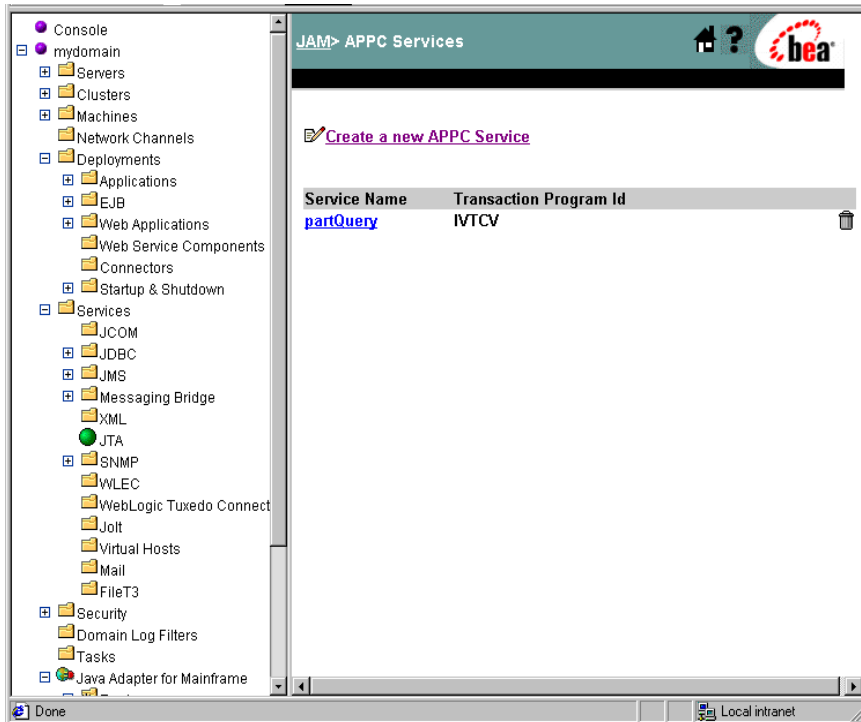
5. Click **Start** to start the Gateway. Click **Stop** to stop the Gateway. The Status field will change accordingly.

# Enabling a Service

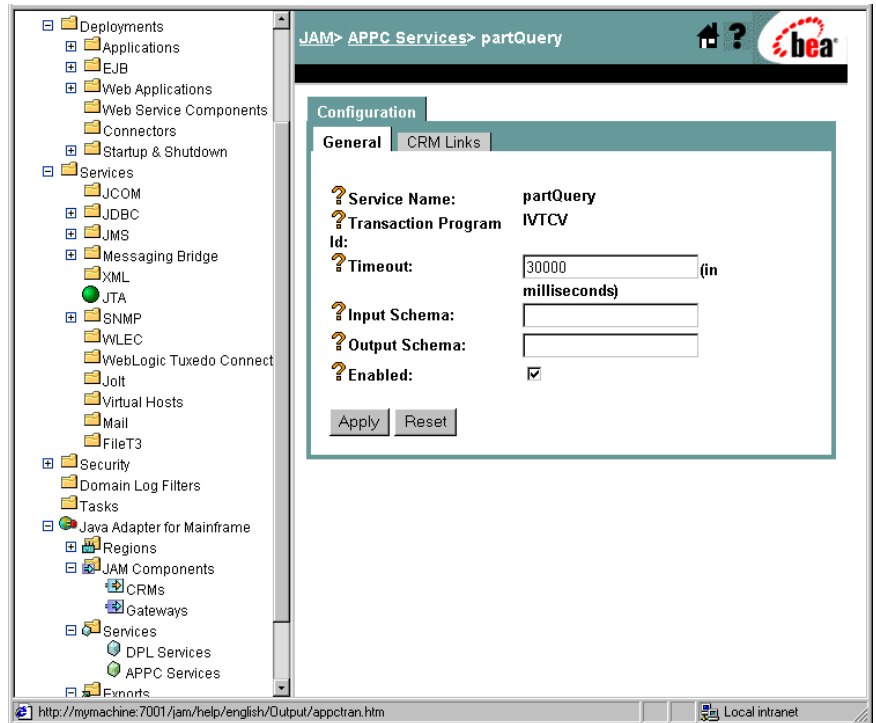
To enable or disable a service:



1. From the WebLogic JAM home page, click the type of service you want to enable or disable. (APPC or DPL)



2. Click the service name you want to enable (or disable).



3. To enable the service, click the Enabled check box so that it is checked.

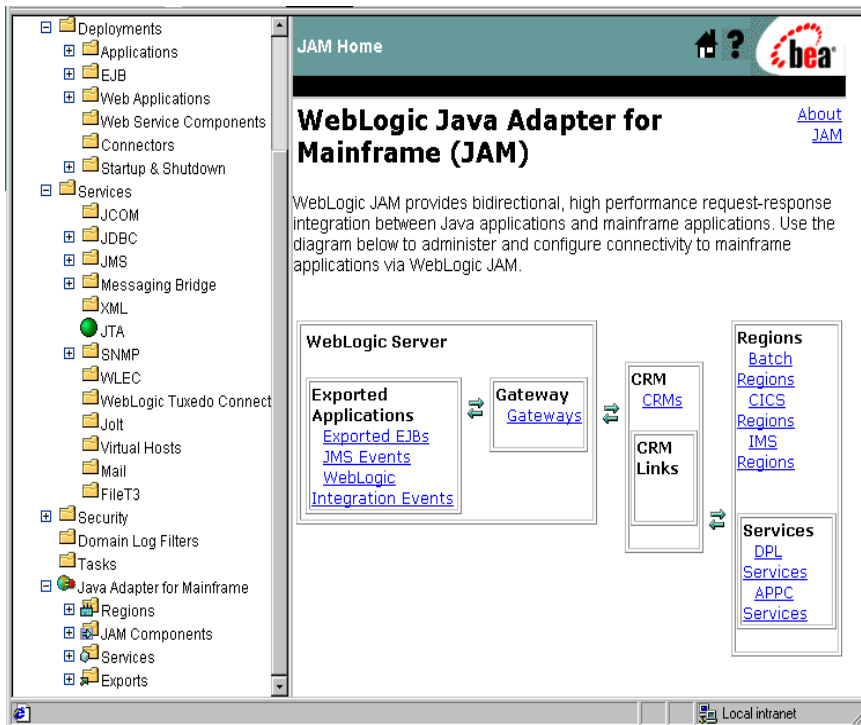
or

To disable the service, click the Enabled check box so that it is not checked.

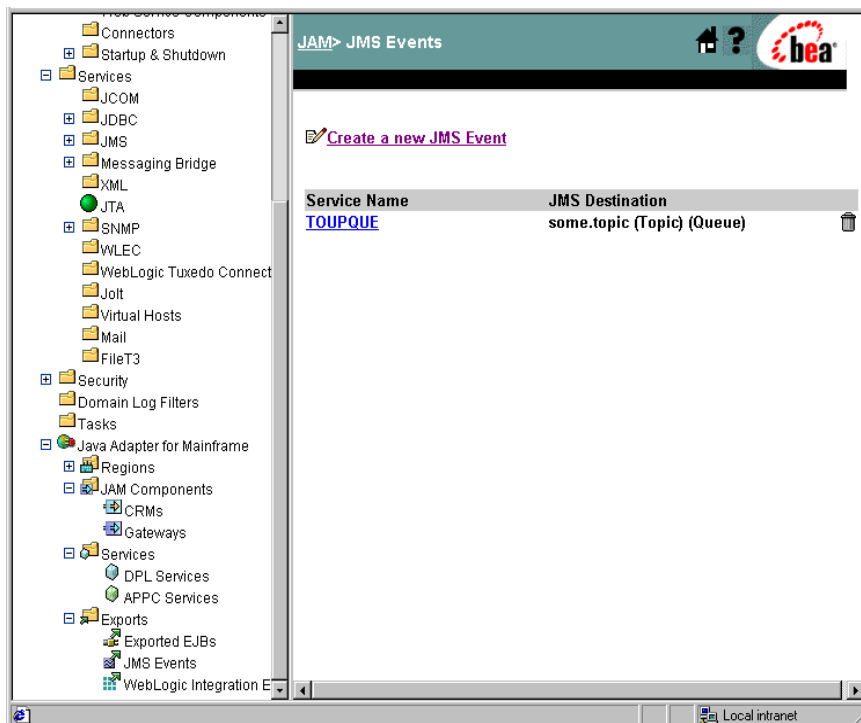
4. Click **Apply**.

# Enabling an Exported WebLogic Application

To enable or disable an exported WebLogic application:

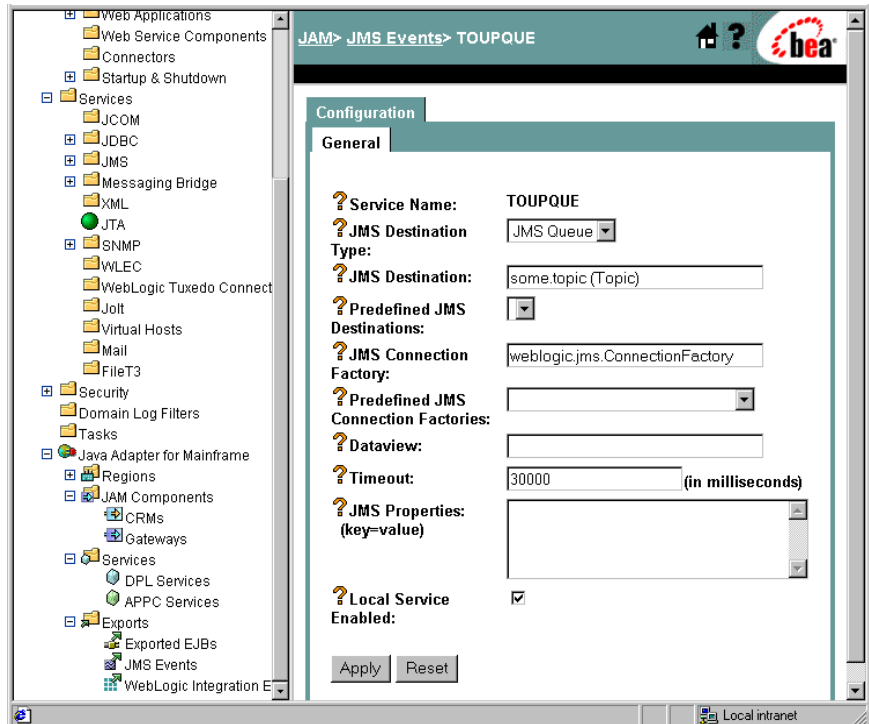


1. From the WebLogic JAM home page, click the type of exported WebLogic application you want to enable. (EJB, JMS, or WebLogic Integration.)





- Click the application name you want to enable or disable.



- To enable the application, click the Local Service Enabled check box so that it is checked.

or

To disable the service, click the Local Service Enabled check box so that it is not checked.

- Click **Apply**.



# 7 CRM Administration

The CRM establishes and monitors communication links between the WebLogic JAM Gateway and mainframe systems, such as CICS, IMS, and batch. The CRM also provides support for the coordination of transactional resources using the two-phase commit protocol. For more information about how the CRM functions within WebLogic JAM, see [“WebLogic JAM Architecture.”](#)

With WebLogic JAM, you may administer and run the CRM in two environments: an environment or a z/OS or OS/390 Unix environment. This section provides information and steps for administering the CRM in both environments.

This section provides information on the following subjects:

- [Starting the CRM](#)
- [Stopping the CRM](#)
- [Activating and De-Activating CRM Links](#)
- [CRM Console Commands](#)

## Starting the CRM

The `CRM` command launches the Communications Resource Manager. You can start the CRM from either a z/OS or OS/390 MVS or z/OS or OS/390 Unix platform. The CRM may be started from the command line or from a shell script. You can enter the `CRM` command from a Unix command line or you may also use JCL on a z/OS or OS/390 MVS platform to run the CRM. Sample JCL is included with your WebLogic JAM installation.

To start the CRM:

1. Determine the required arguments for the CRM command as shown in Table 7-1.

**Table 7-1 Arguments for the CRM command**

Keyword	Default	Optional/ Required	Description
<hostname>: <port>	None	Required	The machine name or dotted-IP address and port number (//hostname:port) of the machine where the CRM is running.
<crm name>	None	Required	The name of the CRM configured in the WebLogic JAM configuration.
-t [0 1 2 3]	0	Optional	<p>Sets the level of tracing.</p> <p><b>0=No tracing.</b></p> <p>Setting this level effectively disables CRM tracing.</p> <p><b>1=Minimum tracing.</b></p> <p>At this level, the CRM traces only major events and is sufficient only to determine the sequence of application conversations.</p> <p><b>2=Medium tracing.</b></p> <p>At this level, the CRM also traces all APPC verbs and the communication between the WebLogic JAM Gateway and the CRM.</p> <p><b>3=Maximum tracing.</b></p> <p>At this level, the CRM also traces all I/O buffers.</p>
-s	disabled (if not present)	Optional	Turns on APPC Stack API traces.

Keyword	Default	Optional/ Required	Description
<code>-p &lt;nbr&gt;</code>	100 threads	Optional	<p>Turns on the performance option and indicates the number of threads used to service requests.</p> <p>This value should correspond to the load of SNA requests that will be made concurrently across a CRM and all of its links. If the number of requests exceeds the number of threads, the request is still executed; however, the completion time could be affected.</p> <p>Do not exceed 200 threads. The CRM is tuned for a maximum of 200 threads. Lower the threads value if you have a restriction on the number of threads that can be active in your system.</p>

2. Follow the instructions for either:

- [Starting the CRM in an MVS Environment](#)

or

- [Starting CRM in z/OS or OS/390 Unix Environment](#)

## Setting CRM Trace Level at CRM Startup

The CRM trace option can be enabled by using the `-t` option. For example, for full tracing:

```
CRM -t3 //myhost:7011 CRMB
```

The list in [Table 7-1](#) describes the three levels of tracing that can be set, and the functionality corresponding to each level.

On z/OS or OS/390 Unix systems, traces are written to a file in the directory in which the CRM was started. If the environment variable `SNACRMWORKDIR` is set, the trace is written to the directory it specifies. The file name is specified as:

```
CRM.<pid>.trace.<seq>
```

Where `<pid>` is the process ID of the CRM process, and `<seq>` is the sequence number of the trace file, which is always 0.

On MVS systems, traces are written to the file identified by `DD:TRACE`.

**Note:** Trace options for the CRM may also be set from the WebLogic Administration Console. See [“Modifying CRM Trace Level Settings”](#) for more information about setting trace options from the WebLogic Administration Console.

### Setting APPC API Tracing at Startup

You can capture the VTAM APPC API by enabling the APPC API tracing. The API trace shows the parameters and values passed and returned to the VTAM APPC stack. The API trace is captured to the GTF tracing facility. The GTF tracing facility must be active in the mainframe region to capture the API traces.

The CRM APPC API trace option can be enabled by using the `-s` option. For example:

```
CRM -s //myhost:7011 CRMB
```

After capturing the traces, you must format the print using GTF formatting procedures such as IPCS. The APPC API trace is written to GTF as user ID `2EA`. You may use this ID to filter the GTF print to include only the APPC API traces.

**Note:** Trace options for the CRM may also be set from the WebLogic Administration Console. See [“Setting APPC API Tracing in the WebLogic Administration Console”](#) for more information about setting trace options from the WebLogic Administration Console.

## Starting the CRM in an MVS Environment

To start the CRM in an MVS environment, you use JCL. A JCL sample is included with your WebLogic JAM installation in member `CRMSTART`.

To start the CRM, follow these steps:

1. Set the following environment variable in the environment where the CRM is started. For each running CRM, you must have a unique `APPDIR` set. For MVS, the `APPDIR` is the high-level qualifier for the data sets created by the CRM. A sample `ENV` file is delivered in the `DATA` library. For each concurrently running CRM, you must have a unique `ENV` file.

```
APPDIR=<High level qualifier for data sets to be created in  
APPDIR>
```

2. Use the set commands shown in [Table 7-2](#) to customize the sample JCL.

Table 7-2 JCL set Commands for the Sample CRMSTART

Command Name	Description
<b>SET STARTCMD</b>	Sets the CRM command line parameters. For example: '-t3 " //<hostname>:<port>" <crm name>'
<b>SET OBJLIB</b>	Indicates the name of the PDSE library where the CRM executable is installed.
<b>SET DATA</b>	Indicates the data set containing the ENVFILE.
<b>SET ENVFILE</b>	Indicates the name of the PDS member that contains the environment variables for the CRM. A sample member, ENV, is delivered with your product.
<b>SET SIZE</b>	Defines the region size for the running CRM task. The recommended setting for this option is 0M to allow the CRM to start up and level out to the size it requires.
<b>SET ENV</b>	Indicates the ENVFILE DD name. This value is pre-set.
<b>SET RUNOPTS</b>	Specifies the optional runtime operands for the Language Environment (LE). This is usually used for debugging. See your system administrator for more information if this option is used.
<b>SET CEE</b>	Specifies the high-level qualifier for the LE runtime library. Language Environment is required to run the CRM.  <b>Note:</b> Uncomment the SET CEE line and tailor the STEPLIB concatenation if these libraries are not in your system link library concatenation.
<b>SET CBC</b>	Specifies the high-level qualifier for the C/C++ runtime library.  <b>Note:</b> Uncomment the SET CBC line and tailor the STEPLIB concatenation if these libraries are not in your system link library concatenation.

3. Run a `crmstart` job using JCL written for your system. [Listing 7-1](#) shows a JCL for the `crmstart` command.

## Listing 7-1 Sample JCL for the CRM Command

---

```

//*****
/* THIS JOB IS USED TO RUN THE CRM PROCESS.          *
/*                                                    *
/* @(#) $Id: crmstart.jcl,v 1.3 2001/05/07 23:41:27 jsmith Exp $ *
/* Copyright (c)2000 BEA Systems, Inc., all rights reserved. *
/******
/* YOU MUST SET THE ENVIRONMENT VARIABLES NEEDED BY CRM      *
/******
/* USE THE SET STATEMENTS TO SET THE APPROPRIATE VALUES      *
/* STARTCMD IS THE CRM COMMAND LINE                            *
/* OBJLIB IS THE LOAD LIBRARY CONTAINING THE PROGRAM EXECUTABLES *
/* DATA IS THE DATASET THAT CONTAINS THE ENVIRONMENT VARIABLES *
/* ENVFILE NAMES THE MEMBER THAT CONTAINS THE ENVIRONMENT VARS  *
/* RUNOPTS SETS ANY DESIRED LE RUNTIME OPTIONS (OPTIONAL)      *
/* SIZE SETS THE REGION SIZE FOR THE CRM PROCESS. 0M SETS NO   *
/* LIMITS ON THE REGION SIZE                                   *
/* TAILOR YOUR JCL FOR THE BELOW IF THESE LIBRARIES ARE NOT    *
/* IN YOUR SYSTEM LINK LOAD LIBRARY CONCATENATION             *
/* CEE IS THE HLQ FOR THE LE RUNTIME LIBRARY                   *
/* CBC IS THE HLQ FOR THE C/C++ RUNTIME LIBRARY                *
/******
// SET STARTCMD=' " //<hostname>:<port>" <crm name>'
// SET OBJLIB=
// SET DATA=
// SET ENVFILE=ENV
// SET RUNOPTS=
// SET SIZE=0M
// SET ENV='ENVAR( "_CEE_ENVFILE=DD:ENV" )'
/* SET CEE=CEE,CBC=CBC
//CRM      EXEC  PGM=CRM,REGION=&SIZE,
//  PARM='POSIX(ON) &ENV &RUNOPTS/&STARTCMD'
//STEPLIB DD  DSN=&OBJLIB,DISP=SHR
/*      DD  DSN=&CEE..SCEERUN,DISP=SHR
/*      DD  DSN=&CBC..SCLBDLL,DISP=SHR
//MSGFILE DD  SYSOUT=*
//TRACE   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//ENV      DD  DSN=&DATA(&ENVFILE),DISP=SHR
//

```

---



## Starting CRM in z/OS or OS/390 Unix Environment

You may start the CRM in either console mode or as a background task. In console mode you may interact with the console to perform or monitor different tasks. In background mode, you must use the `CRMDOWN` command to stop the CRM. This task is described in [“Stopping the CRM in a z/OS or OS/390 Unix Environment.”](#) You must also disable the interactive dialog from `STDIN`.

When you start the CRM in z/OS or OS/390 Unix environment, the CRM command line console prompt displays. You may choose to start the CRM in background mode, in which case, no console prompt displays.

To start the CRM in z/OS or OS/390 Unix:

1. Set the following environment variables in the environment where the CRM is started.
  - `PATH` - the `PATH` variable must be modified to include the `bin` subdirectory of your CRM installation directory.
  - `LIBPATH` - the `LIBPATH` variable must be modified to include the `lib` subdirectory of your CRM installation directory.
  - `APPDIR` must be set to the application directory. For Unix, the `APPDIR` indicates the directory for the files created by the CRM.

A sample file, `uss.env`, is delivered with the installation of WebLogic JAM.

For example, if you installed the CRM into the directory `/work/bea/crm`, the following commands will set your `PATH` and `LIBPATH` variables appropriately:

```
export PATH=$PATH: /work/bea/crm/bin
export LIBPATH=$PATH:/work/bea/crm/lib
export APPDIR=/home/bea/jam
```

2. Enter the CRM command with your required option arguments on the command line. The CRM command and optional arguments follows:

```
CRM [ -t 0|1|2|3 ] [-s] [-p <nbr>] <hostname>:<port> <crm name>
```

For example: To start the CRM running on the machine `myhost` listening on port `5587` with the CRM name of `MYCRM`, enter the following on the command line. This example disables standard input, pipes the `std.out` and `std.err` to

files, and runs in background mode rather than console mode. The `STDIN` capability is disabled by piping from the `/dev/null` directory.

```
$ CRM //myhost:5587 MYCRM </dev/null >std.out 2>std.err &
```

## Stopping the CRM

The `crmdown` command shuts down a CRM. You must explicitly shut down the CRM if the WebLogic JAM Gateway is stopped and the CRM needs to be stopped. You can stop the CRM in a z/OS or OS/390 MVS environment or in a z/OS or OS/390 Unix environment.

To stop the CRM:

1. Determine the required arguments for the `crmdown` command as shown in Table 7-3.

**Table 7-3 Command Line Options for `crmdown`**

Keyword	Default	Optional/ Required	Description
<code>-n&lt;hostname&gt;: &lt;port&gt;</code>	None	Required	The machine name or dotted-IP address and port number ( <code>//hostname:port</code> ) of the machine where the CRM is running.
<code>-v</code>	Off	Optional	Specifies verbose. Normally the command will not produce extraneous messages, facilitating use in a script.
<code>-i</code>	Off	Optional	Ignores errors.

2. Follow the instructions for either:

- [Stopping the CRM in an MVS Environment](#)

or

- [Stopping the CRM in a z/OS or OS/390 Unix Environment](#)

## Stopping the CRM in an MVS Environment

To stop the CRM in an MVS environment, you use JCL. A JCL sample, `CRMDOWN`, is included with your WebLogic JAM installation.

1. Use the set commands shown in Table 7-4 to customize the sample JCL.

**Table 7-4 JCL set Commands for the Sample CRMDOWN**

Command Name	Description
<b>SET STOPCMD</b>	Sets the <code>crmdown</code> command line parameters.
<b>SET OBJLIB</b>	Indicates the name of the PDSE library where the <code>crmdown</code> executable is installed.
<b>SET DATA</b>	Indicates the data set containing the <code>ENVFILE</code> .
<b>SET ENVFILE</b>	Indicates the name of the PDS member that contains the environment variables for the <code>crmdown</code> . A sample member, <code>ENV</code> , is delivered with your product.
<b>SET SIZE</b>	Defines the region size for the running <code>crmdown</code> task.
<b>SET RUNOPTS</b>	Specifies the optional runtime operands for the Language Environment (LE). This is usually used for debugging. See your system administrator for more information to use these options.
<b>SET ENV</b>	Indicates the <code>ENVFILE</code> DD name.
<b>SET CEE</b>	Specifies the high-level qualifier for the LE runtime library. Language Environment is required to run the CRM.  <b>Note:</b> Uncomment the <code>SET CEE</code> line and tailor the <code>STEPLIB</code> concatenation if these libraries are not in your system link library concatenation.
<b>SET CBC</b>	Specifies the high-level qualifier for the C/C++ runtime library.  <b>Note:</b> Uncomment the <code>SET CBC</code> line and tailor the <code>STEPLIB</code> concatenation if these libraries are not in your system link library concatenation.

2. Shut down the CRM using JCL written for your system. Listing 7-2 shows a JCL for the `crmdown` command.

---

**Listing 7-2 Sample JCL for crmdown Command**

---

```
//*****
//* THIS JOB IS USED FOR THE STAND-ALONE COMMAND USED          *
//* TO SHUTDOWN THE CRM PROCESS.                                *
//*                                                            *
//* @(#) $Id: crmdown.jcl,v 1.5 2001/05/07 23:41:27 crount Exp $ *
//* Copyright (c)2000 BEA Systems, Inc., all rights reserved.  *
//*****
//* YOU MUST SET THE ENVIRONMENT VARIABLES NEEDED BY CRMDOWN  *
//*****
**
//* STOPCMD INDICATES THE COMMAND LINE FOR CRMDOWN            *
//* OBJLIB IS THE LOAD LIBRARY CONTAINING THE PROGRAM EXECUTABLES *
//* RUNOPTS SETS ANY DESIRED LE RUNTIME OPTIONS (OPTIONAL)      *
//* DATA IS THE DATASET THAT CONTAINS THE ENVIRONMENT VARIABLES *
//* ENVFILE NAMES THE MEMBER THAT CONTAINS THE ENVIRONMENT VARS *
//* SIZE SETS THE REGION SIZE FOR THE CRM PROCESS              *
//**
//* TAILOR YOUR JCL FOR THE BELOW IF THESE LIBRARIES ARE NOT    *
//* IN YOUR SYSTEM LINK LOAD LIBRARY CONCATENATION              *
//* CEE IS THE HLQ FOR THE LE RUNTIME LIBRARY                   *
//* CBC IS THE HLQ FOR THE C/C++ RUNTIME LIBRARY                *
//*****
// SET STOPCMD='-n<host name>:<port> '
// SET OBJLIB=
// SET RUNOPTS=
// SET DATA=
// SET ENVFILE=ENV
// SET SIZE=1M
// SET ENV='ENVAR( "_CEE_ENVFILE=DD:ENV" ) '
// * SET CEE=CEE,CBC=CBC
//CRMDOWN EXEC PGM=CRMDOWN,REGION=&SIZE,
// PARM='POSIX(ON) &ENV &RUNOPTS/&STOPCMD'
//STEPLIB DD DSN=&OBJLIB,DISP=SHR
// * DD DSN=&CEE..SCEERUN,DISP=SHR
// * DD DSN=&CBC..SCLBDLL,DISP=SHR
//ENV DD DSN=&DATA(&ENVFILE),DISP=SHR
//MSGFILE DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//
```

---

## Stopping the CRM in a z/OS or OS/390 Unix Environment

You can stop the CRM in a z/OS or OS/390 Unix environment by running `crmdown` from a command line. If the command could not be successfully sent to the CRM, `crmdown` prints an error message and exits with error code 1. Upon successful completion, `crmdown` exits with exit code 0.

To stop the CRM from a command line:

1. Enter the `crmdown` command with your required option arguments on the command line.

For example: To stop the CRM running on `mach1` at port 5000:

```
crmdown -n mach1:5000
```

## Activating and De-Activating CRM Links

The `crmlkon` and `crmlkoff` commands are used to activate or de-activate CRM links. You can activate and de-activate links from either a z/OS or OS/390 MVS or z/OS or OS/390 Unix platform

**Note:** You can also start and stop a CRM link from the WebLogic Administration Console. For instructions, see [“Modifying CRM Link Definitions.”](#)

This section provides information on the following subjects:

- [Activating CRM Links](#)
- [De-Activating CRM Links](#)

## Activating CRM Links

The `crmlkon` command is responsible for activating CRM links. You can activate CRM links in the z/OS or OS/390 MVS environment or the z/OS or OS/390 Unix environment. This is useful if one or more individual links needs to be restarted after the CRM is connected.

To activate CRM links in any environment:

1. Determine the required arguments for the `crmlkon` command as shown in Table 7-5.

**Table 7-5 Command Line Options for `crmlkon`**

Keyword	Default	Optional/ Required	Description
<code>-n&lt;hostname&gt;: &lt;port&gt;</code>	None	Required	The machine name or dotted-IP address and port number ( <code>//hostname:port</code> ) of the machine where the CRM is running.
<code>&lt;linkname&gt;</code>	None	Required	Names the link or links to be started. Use a space to separate multiple link names.
<code>-v</code>	Off	Optional	Specifies verbose. Normally the command will not produce extraneous messages, facilitating use in a script.
<code>-i</code>	Off	Optional	Ignores errors. When specifying multiple links, any error encountered when issuing CRM commands causes <code>crmlkon</code> to stop processing links and return. This allows errors to be ignored for individual links and processing continues with the next named link.

2. Follow the instructions for either:

- [Activating CRM Links in an MVS Environment](#)

or

- [Activating CRM Links from a z/OS or OS/390Unix Environment](#)

## Activating CRM Links in an MVS Environment

To activate CRM links in an MVS environment, you use JCL. A JCL sample, `crmlkon`, is included with your WebLogic JAM installation. Refer to your system administrator for more information about your particular setup.

To activate CRM links in an MVS environment, follow these steps:

1. Use the set commands shown in Table 7-6 to customize the sample JCL.

**Table 7-6 JCL set Commands for the Sample CRMLKON**

Command Name	Description
<b>SET LINKCMD</b>	Sets the <code>crmlkon</code> command line parameters.
<b>SET OBJLIB</b>	Indicates the name of the PDSE library where the <code>crmlkon</code> executable is installed.
<b>SET DATA</b>	Indicates the data set containing the <code>ENVFILE</code> .
<b>SET ENVFILE</b>	Indicates the name of the PDS member that contains the environment variables for the <code>CRMLKON</code> . A member, <code>ENV</code> , is delivered with your product.
<b>SET SIZE</b>	Defines the region size for the running <code>crmlkon</code> task.
<b>SET ENV</b>	Indicates the <code>ENVFILE</code> DD name. This value is pre-set.
<b>SET RUNOPTS</b>	Specifies the optional runtime operands for the Language Environment (LE). This is usually used for debugging. See your system administrator for more information.
<b>SET CEE</b>	<p>Specifies the high-level qualifier for the LE runtime library. Language Environment is required to run the CRM.</p> <p><b>Note:</b> Uncomment the <code>SET CEE</code> line and tailor the <code>STEPLIB</code> concatenation if these libraries are not in your system link library concatenation.</p>
<b>SET CBC</b>	<p>Specifies the high-level qualifier for the C/C++ run-time library.</p> <p><b>Note:</b> Uncomment the <code>SET CBC</code> line and tailor the <code>STEPLIB</code> concatenation if these libraries are not in your system link library concatenation.</p>

2. Run a crmlkon job using JCL written explicitly for your system. Listing 7-3 shows a JCL for the crmlkon command.

### Listing 7-3 Sample JCL for crmlkon Command

---

```
//*****
**
/** THIS JOB IS USED FOR THE STAND-ALONE LINK COMMAND          *
/** TO ACTIVATE A REMOTE LINK.                                *
/**                                                            *
/** @(#) $Id: crmlkon.jcl,v 1.10 2001/05/07 23:41:27 crount Exp $ *
/** Copyright (c)2000 BEA Systems, Inc., all rights reserved.  *
/*******
/** YOU MUST SET THE ENVIRONMENT VARIABLES NEEDED BY CRMLKON  *
/*******
/*******
/** LINKCMD INDICATES THE DISTRIBUTED CRM ADDRESS AND LINKNAME *
/** OBJLIB IS THE LOAD LIBRARY CONTAINING THE EAM PROGRAM OBJECTS*
/** RUNOPTS SETS ANY DESIRED LE RUNTIME OPTIONS                *
/** DATA IS THE DATASET THAT CONTAINS THE ENVIRONMENT VARIABLES *
/** ENVFILE NAMES THE MEMBER THAT CONTAINS THE ENVIRONMENT VARS *
/** SIZE SETS THE REGION SIZE FOR THE CRM PROCESS              *
/** TAILOR YOUR JCL FOR THE BELOW IF THESE LIBRARIES ARE NOT    *
/** IN YOUR SYSTEM LINK LOAD LIBRARY CONCATENATION             *
/** CEE IS THE HLQ FOR THE LE RUNTIME LIBRARY                  *
/** CBC IS THE HLQ FOR THE C/C++ RUNTIME LIBRARY               *
/*******
// SET LINKCMD='-n<host name>:<port> <linkname>'
// SET OBJLIB=
// SET RUNOPTS=
// SET DATA=
// SET ENVFILE=ENV
// SET SIZE=1M
// SET ENV='ENVAR(" _CEE_ENVFILE=DD:ENV" )'
/** SET CEE=CEE,CBC=CBC
//CRMLKON EXEC PGM=CRMLKON,REGION=&SIZE,
// PARM='POSIX(ON) &ENV &RUNOPTS/&LINKCMD'
//STEPLIB DD DSN=&OBJLIB,DISP=SHR
/** DD DSN=&CEE..SCEERUN,DISP=SHR
/** DD DSN=&CBC..SCLBDLL,DISP=SHR
//ENV DD DSN=&DATA(&ENVFILE),DISP=SHR
//MSGFILE DD SYSOUT=*
//SYSPRINT DD SYSOUT=*

//
```

---



## Activating CRM Links from a z/OS or OS/390Unix Environment

The `crmlkon` command starts all of the CRM links named on the command line. If the command could not be successfully sent to the CRM, `crmlkon` prints an error message and exits with error code 1. Upon successful completion, `crmlkon` exits with exit code 0.

To activate CRM links from a Unix command line:

1. Enter the `crmlkon` command with your required option arguments.

For example, to start links, `link2` and `cicstest`, owned by the CRM running on `mach1` at port 5000:

```
crmlkon -n mach1:5000 link2 cicstest
```

## De-Activating CRM Links

The `crmlkoff` command is responsible for de-activating CRM links. You can de-activate CRM links in the z/OS or OS/390 MVS environment or the z/OS or OS/390 Unix environment. This is useful if one or more individual links needs to be stopped after the CRM is connected.

To de-activate CRM links in any environment:

1. Determine the required arguments for the `crmlkoff` command as shown in Table 7-7.

**Table 7-7 Command Line Options for `crmlkoff`**

Keyword	Default	Optional/ Required	Description
<code>-n&lt;hostname&gt;: &lt;port&gt;</code>	None	Required	The machine name or dotted-IP address and port number ( <code>//hostname:port</code> ) of the machine where the CRM is running.
<code>&lt;linkname&gt;</code>	None	Required	Names the link or links to be started. Use a space to separate multiple link names.

Keyword	Default	Optional/ Required	Description
-v	Off	Optional	Specifies verbose. Normally the command will not produce extraneous messages, facilitating use in a script.
-i	Off	Optional	Ignores errors. When specifying multiple links, any error encountered when issuing CRM commands causes <code>crmlkon</code> to stop processing links and return. Errors can be ignored for individual links and processing continues with the next named link

2. Follow the instructions for either:

- [De-Activating a CRM Link in an MVS Environment](#)

or

- [De-Activating a CRM Link in a z/OS or OS/390 Unix Environment](#)

## De-Activating a CRM Link in an MVS Environment

To de-activate CRM links in an MVS environment, you use JCL. A JCL sample, `crmlkoff`, is included with your WebLogic JAM installation.

1. Use the set commands shown in Table 7-8 to customize the sample JCL.

**Table 7-8 JCL set Commands for the Sample CRMLKOFF**

Command Name	Description
<b>SET LINKCMD</b>	Sets the <code>crmlkoff</code> command line parameters.
<b>SET OBJLIB</b>	Indicates the name of the PDSE library where the <code>crmlkoff</code> executable is installed.
<b>SET RUNOPTS</b>	Specifies the optional runtime operands for the Language Environment (LE). This is usually used for debugging. See your system administrator for more information if using these options.

Command Name	Description
<b>SET DATA</b>	Indicates the data set containing the ENVFILE.
<b>SET ENVFILE</b>	Indicates the name of the PDS member that contains the environment variables for the <code>crmlkoff</code> . A member, ENV, is delivered with your product.
<b>SET SIZE</b>	Defines the region size for the running <code>crmlkoff</code> task.
<b>SET ENV</b>	Indicates the ENVFILE DD name. This value is pre-set.
<b>SET CEE</b>	Specifies the high-level qualifier for the LE runtime library. Language Environment is required to run the CRM.  <b>Note:</b> Uncomment the SET CEE line and tailor the STEPLIB concatenation if these libraries are not in your system link library concatenation.
<b>SET CBC</b>	Specifies the high-level qualifier for the C/C++ run-time library.  <b>Note:</b> Uncomment the SET CBC line and tailor the STEPLIB concatenation if these libraries are not in your system link library concatenation.

- De-activate a CRM link using JCL written explicitly for your system. Listing 7-4 shows a sample JCL for the `crmlkoff` command.

### Listing 7-4 Sample JCL for crmlkoff Command

```

/*****
/* THIS JOB IS USED FOR THE STAND-ALONE LINK COMMAND          *
/* TO DEACTIVATE A REMOTE LINK.                                *
/*                                                             *
/* @(#) $Id: crmlkoff.jcl,v 1.10 2001/05/07 23:41:27 crount Exp $
/* Copyright (c)2000 BEA Systems, Inc., all rights reserved.   *
/*****
/* YOU MUST SET THE ENVIRONMENT VARIABLES NEEDED BY CRMLKOFF  *
/*****
/*****
/* LINKCMD INDICATES THE DISTRIBUTED CRM ADDRESS AND LINKNAME  *
/* OBJLIB IS THE LOAD LIBRARY CONTAINING THE EAM PROGRAM OBJECTS*
/* RUNOPTS SETS ANY DESIRED LE RUNTIME OPTIONS                *

```

```

/** DATA IS THE DATASET THAT CONTAINS THE ENVIRONMENT VARIABLES *
/** ENVFILE NAMES THE MEMBER THAT CONTAINS THE ENVIRONMENT VARS *
/** SIZE SETS THE REGION SIZE FOR THE CRM PROCESS *
/** TAILOR YOUR JCL FOR THE BELOW IF THESE LIBRARIES ARE NOT *
/** IN YOUR SYSTEM LINK LOAD LIBRARY CONCATENATION *
/** CEE IS THE HLQ FOR THE LE RUNTIME LIBRARY *
/** CBC IS THE HLQ FOR THE C/C++ RUNTIME LIBRARY *
/*******
// SET LINKCMD='-n<host name>:<port> <linkname>'
// SET OBJLIB=
// SET RUNOPTS=
// SET DATA=
// SET ENVFILE=ENV
// SET SIZE=1M
// SET ENV='ENVAR( "_CEE_ENVFILE=DD:ENV" )'
/** SET CEE=CEE,CBC=CBC
//CRMLKOFF EXEC PGM=CRMLKOFF,REGION=&SIZE,
// PARM=' POSIX(ON) &ENV &RUNOPTS/&LINKCMD'
//STEPLIB DD DSN=&OBJLIB,DISP=SHR
/** DD DSN=&CEE..SCEERUN,DISP=SHR
/** DD DSN=&CBC..SCLBDLL,DISP=SHR
//ENV DD DSN=&DATA(&ENVFILE),DISP=SHR
//MSGFILE DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//

```

---

### De-Activating a CRM Link in a z/OS or OS/390 Unix Environment

The `crmlkoff` command stops all of the CRM links named on the command line. If the command could not be successfully sent to the CRM, `crmlkoff` prints an error message and exits with error code 1. Upon successful completion, `crmlkoff` exits with exit code 0.

To de-activate CRM links from a Unix command line:

1. Enter the `crmlkoff` command with your required option arguments.

For example: To stop links `link1` and `cicstest` owned by the CRM running on `mach` at port 5000:

```
crmlkoff -n mach:5000 link1 cicstest
```

# CRM Console Commands

When you run the CRM from the z/OS or OS/390 Unix command line, you are presented with a CRM prompt. Table 7-9 lists the commands the you can enter at the CRM prompt and the tasks to perform. Some of the functionality for these commands is also available from the WebLogic Administration Console. For these cases, the corresponding location in the WebLogic Administration Console is provided.

**Table 7-9 CRM Commands and Corresponding WebLogic Administration Console Functions**

CRM Console Commands	CRM Task	WebLogic Administration Console Location
DA	Display All Tasks	None
DL	Display Link Status	JAM → CRM → <CRM Name > → CRM Links
DS	Display Link Statistics	JAM → CRM → <CRM Name > → CRM Links → <CRM Link Name>
DT	Display Trace Status	JAM → CRM → <CRM Name > → Administration Tab → Diagnostics Tab
LS	Stop Link(s)	JAM → CRM → <CRM Name > → Administration Tab → Start/Stop Tab
SL	Start Link(s)	JAM → CRM → <CRM Name > → Administration Tab → Start/Stop Tab
ST	Start All Links	JAM → CRM → <CRM Name > → Administration Tab → Start/Stop Tab
SH	Shutdown (normal)	None
SI	Shutdown Immediate	None
DC	Disconnect	None

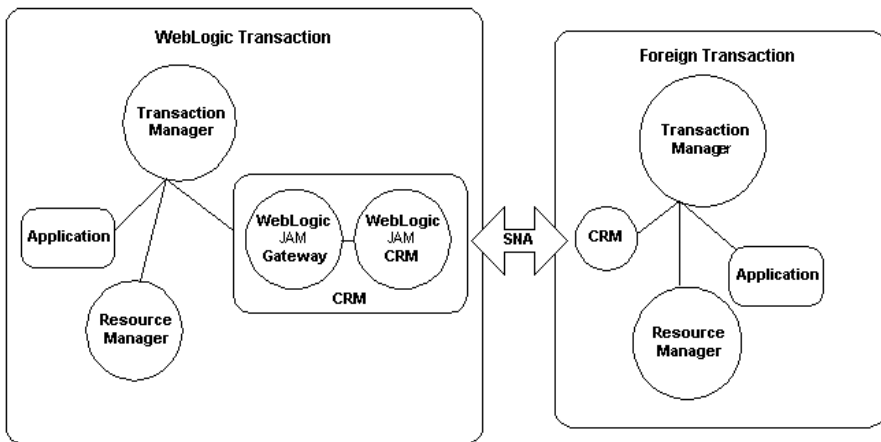
CRM Console Commands	CRM Task	WebLogic Administration Console Location
DF	Display Flags	JAM → CRM → <CRM Name > → Administration Tab → Start/Stop Tab
AY	Enable APPC Trace	JAM → CRM → <CRM Name > → Administration Tab → Diagnostics Tab
AN	Disable APPC Trace	JAM → CRM → <CRM Name > → Administration Tab → Diagnostics Tab
T0	Set Trace Level to 0	JAM → CRM → <CRM Name > → Administration Tab → Diagnostics Tab
T1	Set Trace Level to 1	JAM → CRM → <CRM Name > → Administration Tab → Diagnostics Tab
T2	Set Trace Level to 2	JAM → CRM → <CRM Name > → Administration Tab → Diagnostics Tab
T3	Set Trace Level to 3	JAM → CRM → <CRM Name > → Administration Tab → Diagnostics Tab
L0	Stop Link	JAM → CRM → <CRM Name > → Administration Tab → Start/Stop Tab
L1	Start Link	JAM → CRM → <CRM Name > → Administration Tab → Start/Stop Tab
NLS	Named Link Stop	JAM → CRM → <CRM Name > → Administration Tab → Start/Stop Tab
SNL	Start Named Link	JAM → CRM → <CRM Name > → Administration Tab → Start/Stop Tab

# 8 Administering Transactions

WebLogic Server provides J2EE transactional support using Java Transactional API (JTA). WebLogic Server is the transaction manager. The WebLogic JAM Gateway will register with the transaction manager, and become part of a transaction. When a request is made to a remote service, a foreign transaction manager becomes involved in the distributed transaction. The foreign transaction manager is one of the following:

- CICS or CICS using RRS
- IMS using APPC/MVS and RRS
- Batch processes using APPC/MVS and RRS

**Figure 8-1 Transactional Support**



The two-phase commit protocol is a method of coordinating a single transaction across two or more resource managers, ensuring data integrity. A single transaction is coordinated by a transaction manager. When the transaction spans transaction managers, then an additional resource manager is required, the communications resource manager. The communication resource manager manages the two phase commit protocol to the foreign transaction manager. This is referred to as a distributed WebLogic Server transaction.

# Configuring Logical Units with Transactional Support

When WebLogic JAM is started, it will always attempt to configure the mainframe domain as part of the distributed WebLogic Server transaction environment. If a CRM link is not configured to support transactions, WebLogic JAM will establish on startup that the link will not support transactions.

If the following message appears during link startup, that link is not configured to support transactions.

```
"WARN: Synclevl on link <link> negotiated down to 1"
```

In order for the link between the CRM and the back-end environments to support transactions, both environments must be configured for synclevel support.

## Configure the CRM Logical Unit

To configure the CRM environment for synclevel, the CRM logical unit must be configured. The `SYNCLVL` parameter on the VTAM `APPLID` definition must be set to `SYNCPT`. For further information about setting VTAM `APPLID` parameters, refer to [“Configuring WebLogic JAM Connectivity.”](#)



## Configure the Back-end Environment

The back-end application must be configured to support synclevel.

- CICS environment

Transactions are supported across a link by default. There is no special configuration required.

- IMS environment

The APPC/MVS logical unit must be configured. The `SYNCLVL` parameter on the VTAM APPLID definition must be set to `SYNCPT`. The `ATTNLOSS` parameter must be set to `ALL`.

- Batch environment

The APPC/MVS logical unit must be configured. The `SYNCLVL` parameter on the VTAM APPLID definition must be set to `SYNCPT`. The `ATTNLOSS` parameter must be set to `ALL`.

For each of these environments, if RRS is used as the transaction manager, RRS must be active. However, if RRS is not active, there will be no notification when the link is started that the environment cannot support transactions.

## Transaction Log Files

To ensure data integrity, a transaction manager maintains a transaction log. Each log entry tracks the state of a two-phase commit sequence. When two transaction managers are involved, the logs of both managers must be synchronized. In the SNA protocol, two transaction managers synchronize their logs by exchanging and comparing log names when a link is started or restarted.

In addition, the WebLogic JAM CRM maintains its own transaction logs.

### JTA Log Files

WebLogic Server log files are stored under the `config/<domain>/logs` directory. The WebLogic administration console can be used to view these logs.

### WebLogic JAM CRM Log Files

WebLogic JAM CRM log files are stored under the application directory of the CRM. The location of these files is determined by the setting of the APPDIR environment variable. APPDIR can specify a directory in a z/OS or OS/390 Unix environment, or a high-level qualifier of the log data set in the MVS environment.

The JAM CRM creates two types of log files, restart logs and blob logs. Both can be viewed using the CRMLOGS utility.

#### Restart Log File

The restart log stores a single entry for each link. Each entry contains the log names for the CRM and the back-end system for one link. These names are used to perform log comparisons between the two.

The file name is described as follows:

CRM.<crm name>.RSTRTLOG

**Note:** On an MVS system, there is a high-level qualifier preceding the file name.

#### Blob Log File

The blob log file maintains the state of the two-phase transmit protocol for each active transaction. The file name is described as follows:

CRM.<crm name>.BLOBLOG

**Note:** On an MVS system, there is a high-level qualifier preceding the file name.

## Viewing CRM Log Files

Use the CRMLOGS command to view the contents of the CRM restart log and blob log. This is useful if in-flight transactions have been stopped and need to be recovered. For more information about recovering transactions, see [“Recovery.”](#)

To view the CRM log files:

1. Determine the required arguments for the CRMLOGS command as shown in Table 8-1.

**Table 8-1 Command Line Option for CRMLOGS**

Keyword	Default	Optional/ Required	Description
<crm name>	None	Required	The CRM name used on the CRM command line, and configured in the WebLogic Administration Console.

2. Follow the instructions for either:

- [Viewing the CRM Log Files in an MVS Environment](#)

or

- [Viewing CRM Log Files in an z/OS or OS/390 Unix Environment](#)

### Viewing the CRM Log Files in an MVS Environment

To view the CRM log files in an MVS environment, use JCL. A sample JCL is included with your WebLogic JAM installation in member.

1. Use the set commands shown in Table 8-2 to customize the sample JCL.

**Table 8-2 JCL set Commands for the Sample CRMLOGS**

Command Name	Description
<b>SET LOGSCMD</b>	Sets the CRMLOGS command line parameters.

Command Name	Description
<b>SET OBJLIB</b>	Indicates the name of the PDSE library where the CRMLOGS executable is installed.
<b>SET RUNOPTS</b>	Specifies the optional runtime operands for the Language Environment (LE). This is usually used for debugging. See your system administrator for more information.
<b>SET DATA</b>	Indicates the data set containing the ENVFILE.
<b>SET ENVFILE</b>	Indicates the name of the PDS member that contains the environment variables for the CRMLOGS. A sample member, ENV, is delivered with your product.
<b>SET RUNOPTS</b>	Specifies the optional runtime operands for the Language Environment (LE). This is usually used for debugging. See your system administrator for more information if this option is used.
<b>SET SIZE</b>	Defines the region size for the running CRMLOGS task.
<b>SET ENV</b>	Indicates the ENVFILE DD name. This value is pre-set.
<b>SET CEE</b>	Specifies the high-level qualifier for the LE run-time library.  <b>Note:</b> Uncomment the SET CEE line and tailor the STEPLIB concatenation if these libraries are not in your system link library concatenation.
<b>SET CBC</b>	Specifies the high-level qualifier for the C/C++ runtime library.  <b>Note:</b> Uncomment the SET CBC line and tailor the STEPLIB concatenation if these libraries are not in your system link library concatenation.

2. Run a CRMLOGS job using JCL written for your system. [Listing 8-1](#) shows a JCL for the CRMLOGS command.

### Listing 8-1 Sample JCL for CRMLOGS Command

```
// *****
**
// * THIS JOB IS USED TO CHECK THE RECOVERY LOGS FOR *
// * OUTSTANDING TRANSACTION DATA. *

```

```

/*                                                                 *
/** @(#) $Id: crmlogs.jcl,v 1.3 2000/05/17 16:20:27 jsmith Exp $   *
/** Copyright (c)2000 BEA Systems, Inc., all rights reserved.    *
/**                                                                 *
/** *****                                                        *
/** YOU MUST SET THE ENVIRONMENT VARIABLES NEEDED BY CRMLOGS     *
/** *****                                                        *
/** *****                                                        *
/** LOGSCMD IS USED TO SET THE DESIRED CRM NAME                   *
/** OBJLIB IS THE LOAD LIBRARY CONTAINING THE PROGRAM OBJECTS    *
/** RUNOPTS SETS ANY DESIRED LE RUNTIME OPTIONS (OPTIONAL)        *
/** DATA IS THE DATA SET THAT CONTAINS THE ENVIRONMENT VARIABLES *
/** ENVFILE NAMES THE MEMBER THAT CONTAINS THE ENVIRONMENT VARS   *
/** SIZE SETS THE REGION SIZE FOR THE CRM PROCESS                *
/** ENV SETS THE ENVIRONMENT VARIABLES DD NAME                    *
/** TAILOR YOUR JCL FOR THE BELOW IF THESE LIBRARIES ARE NOT     *
/** IN YOUR SYSTEM LINK LOAD LIBRARY CONCATENATION               *
/** CEE IS THE HLQ FOR THE LE RUNTIME LIBRARY                    *
/** CBC IS THE HLQ FOR THE C/C++ RUNTIME LIBRARY                  *
/** *****                                                        *
/** SET LOGSCMD=<crm name>                                        *
/** SET OBJLIB=                                                  *
/** SET DATA=                                                    *
/** SET ENVFILE=                                                  *
/** SET RUNOPTS=                                                  *
/** SET SIZE=10M                                                  *
/** SET ENV='ENVAR( "_CEE_ENVFILE=DD:ENV" ) '                    *
/** SET CEE=CEE,CBC=CBC                                           *
/**CRMLOGS EXEC PGM=CRMLOGS,REGION=&SIZE,                          *
/** PARM='POSIX(ON) &ENV &RUNOPTS/&SNAGRP'                        *
/**STEPLIB DD DSN=&OBJLIB,DISP=SHR                                *
/** DD DSN=&CEE..SCEERUN,DISP=SHR                                  *
/** DD DSN=&CBC..SCLBDLL,DISP=SHR                                  *
/**MSGFILE DD SYSOUT=*                                           *
/**SYSPRINT DD SYSOUT=*                                           *
/**ENV DD DSN=&DATA(ENVFILE),DISP=SHR                             *
/**

```

## Viewing CRM Log Files in an z/OS or OS/390 Unix Environment

You can view the CRM log files in a z/OS or OS/390 Unix environment by running CRMLOGS from a command line. To view the CRM log files from a command line enter the CRMLOGS command with the CRM name.

For example, to view CRM logs for CRM `mycrm`, enter the following at the command line.

```
CRMLOGS mycrm
```

## Mainframe Application Log

Each type of back-end environment maintains a transaction log, and has established methods for viewing the log. For information on maintaining and viewing these logs, refer to the product documentation for your environment.

## Restarting Links (cold/warm)

When restarting a transactional link, all parties involved in the distributed transaction must agree upon the integrity of this restart. This includes the identification required for SNA communication, and agreement by all parties on the recovery of any items that were in two-phase commit sequence when the link was stopped.

This information is kept in the logs that were described in the previous section, and include the WebLogic Server logs, the mainframe application logs and the WebLogic JAM CRM restart and blob logs.

There are two ways to restart a transactional link with regards to its integrity: cold start and warm start.

- **Cold Start**

Starting / restarting a link without ensuring the integrity of the restart.

- **Warm Start**

Ensuring the integrity of the link before starting/restarting a link.

## Cold Start

When a link is started for the first time, the CRM log files are created. The SNA communication between the WebLogic JAM CRM and the mainframe application (CICS, IMS or APPC/MVS) requires that both sides exchange and hold on to their log names. The CRM creates a restart log to hold this information. The information must match the information that is held by the remote mainframe region on each subsequent restart. In addition, the CRM creates a blob log to hold all two-phase commit sequence information for each transactional request.

When a link must be restarted, but does not require preserving the integrity of the previous start, (the log names will not be exchanged and no items will be recovered), then the link may be started with a cold start.

### Starting a Link with a Cold Start

The following steps are required to start the link with a cold start:

- Remove the WebLogic transaction logs.

**Warning:** This will affect all in-flight transactions within the WebLogic Server domain.

- Remove the CRM blob and restart logs.

**Warning:** This will affect the start requirements for all links configured for the CRM.

- Restart the mainframe region with a cold start, or maintain the mainframe application logs so that the link may be restarted cold.

**Warning:** Check with your administrator on how this will affect the other resources in the region.

Although a cold start is not recommended for a restart of a link, there are times when a cold start is necessary. Usually this is only done when manual intervention was used to recover in-flight transactional requests on the link, or when there is no recovery required to restart the link.

### Warm Start

A warm start occurs when a link is restarted while ensuring the integrity of the previous start. The SNA communication between the WebLogic JAM CRM and the mainframe application (CICS, IMS or APPC/MVS) exchange log names to ensure that data integrity can be maintained on the restart. In addition, any in-flight transactions that are recorded in the WebLogic Server transaction logs, the CRM transaction blob log or the mainframe application transaction logs are recovered during the restart procedure.

A warm start is unsuccessful if the CRM link log names and the mainframe application log names are not accepted by either side in the exchange logs sequence. The link will be established, but will be unable to service transactional requests.

If all items in recovery are not recovered on the restart, the link will be established as a transactional link. However, the items that were not recovered remain “unrecovered”, and the resources allocated to the transaction remain unavailable. Manual intervention must occur to recover the items.

### Recovery

The transaction logs in the WebLogic JAM distributed transaction are used to maintain the two-phase commit state of a transactional request. The two-phase commit is the sequence of events that occurs between the transaction managers and resource managers when a commit or rollback begins.

If the transaction should fail before the sequence of events is complete, and the transaction managers are unable to communicate the failure successfully and release the transaction resources, then the transaction must be recovered. This typically happens only when a resource, a resource manager or a transaction manager becomes unavailable. When the unavailable entity becomes available again, then all participants in the transaction attempt to finalize or recover the transaction to its final state.



## Initiating Recovery

To initiate recovery, the failed entity must be restarted, and the participating WebLogic Server or Servers must be stopped and restarted. The actual recovery occurs when all participants agree to commit or rollback the transaction resources and release the transaction resources. If the participants in the transaction are unable to agree how to handle the recovery, then the state of the transaction becomes heuristic. Manual intervention must occur to update the resources appropriately.

**Notes:** Until recovery is performed for a transaction, the transaction resources will be held and remain unavailable. This includes the SNA session resources.

To initiate recovery, the WebLogic Server or Servers which participated in the failed transactions *must* be restarted.



# 9 Preparing Your System for Planned Outages

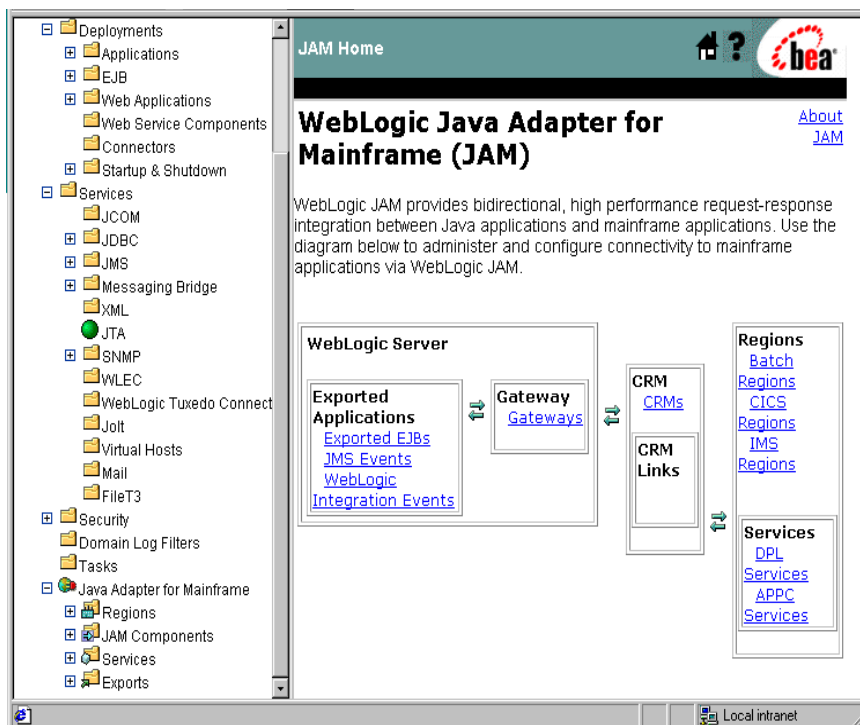
Occasionally, some or all of your WebLogic JAM system may need to be shut down for planned outages. Depending on what hardware or software component is going to be taken out of service, you will need to shut down different parts of your WebLogic JAM system. If you have configured your WebLogic JAM system with redundancy, part of your system can be shut down while the remaining components continue to function.

Depending on the type of outage, a different WebLogic JAM component will have to be shut down. If a remote region is to be taken offline, you should shut down the CRM Links to that region. If a mainframe is to be taken offline, you should shut down any CRM that runs on this mainframe. If an instance of WebLogic Server is to be shut down, you should stop the Gateway that runs in this instance. This section provides information on the following subjects:

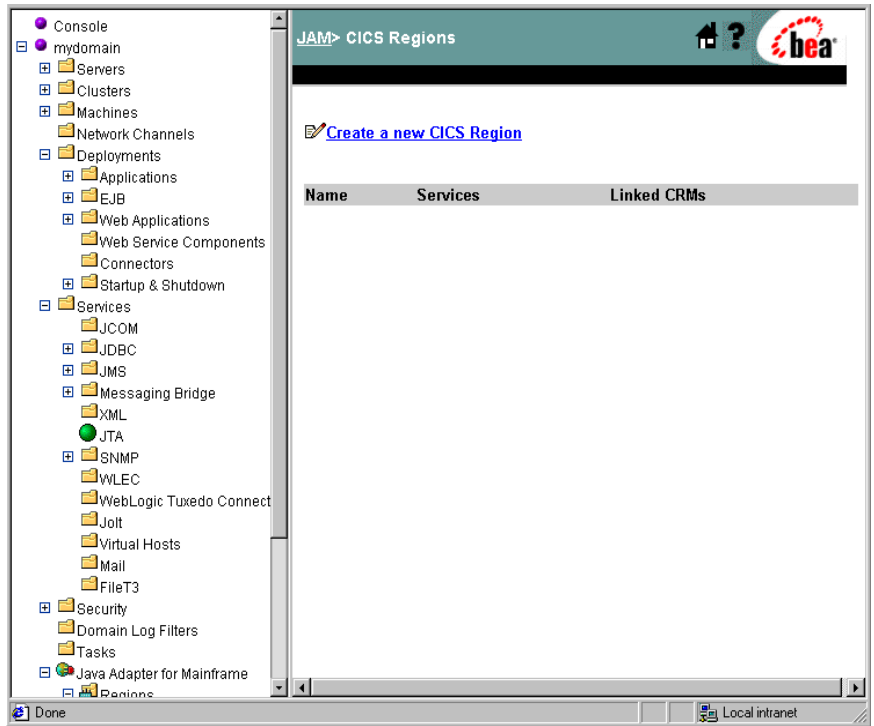
- [Shutting Down a Remote Region](#)
- [Shutting Down a Mainframe](#)
- [Shutting Down an Instance of WebLogic Server](#)

# Shutting Down a Remote Region

1. If a region that your WebLogic JAM system is connected to is going to be taken out of service, you should stop all CRM Links that connect to this region. To see what CRM Links connect to this region from the WebLogic Administration Console, use the following steps:

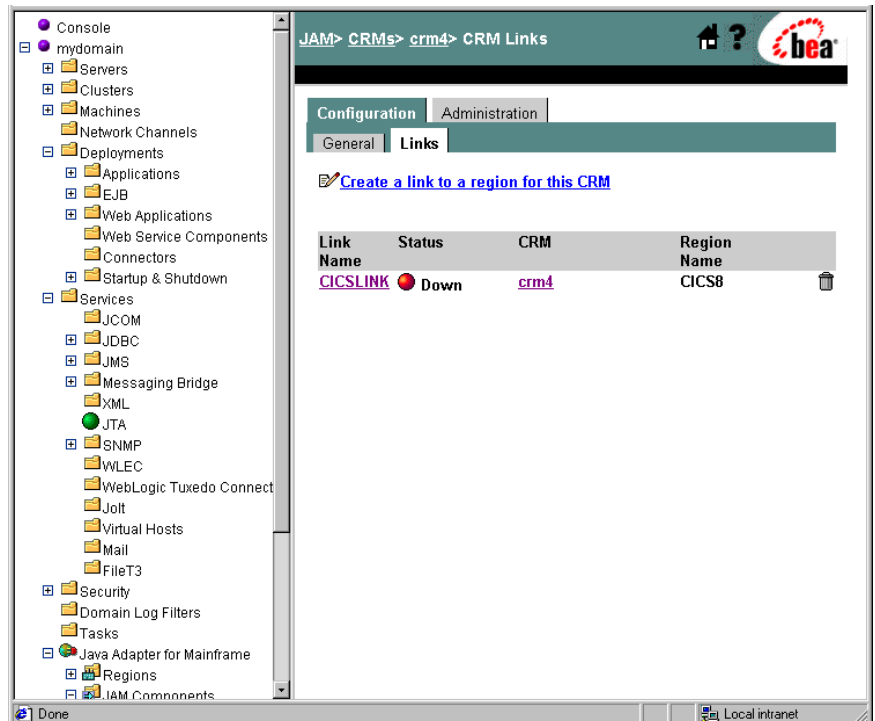


1. From the WebLogic JAM home page, click the type of region (Batch, CICS, IMS) that you want to shut down. The List Regions page displays.

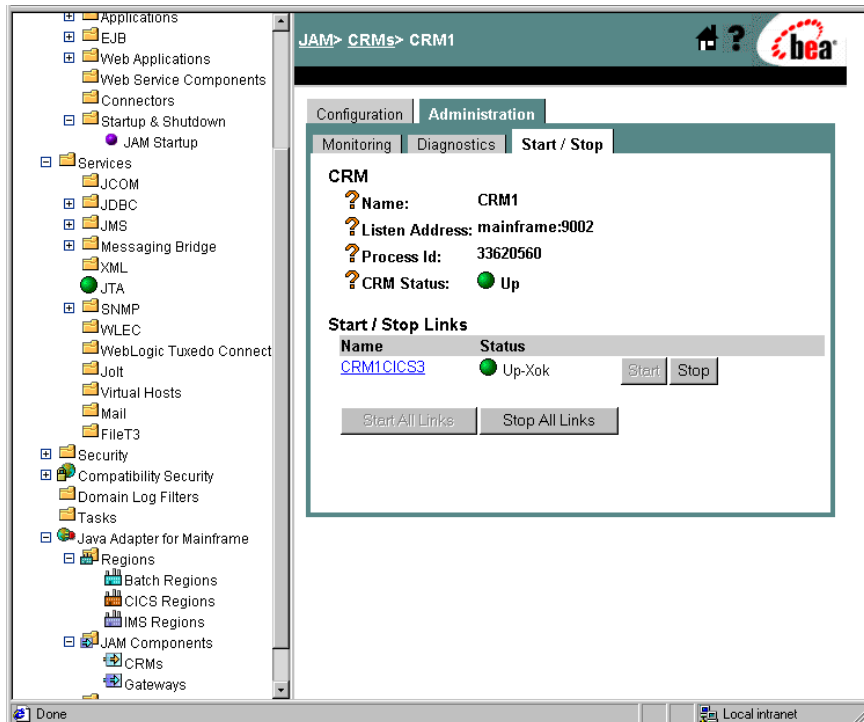


## 9 *Preparing Your System for Planned Outages*

2. In the Linked CRMs column, click the list for the region that is going out of service.



- For each listed CRM link, click the status icon. The Start/Stop Link page displays.



- Click Stop to stop the CRM Link.
- Click the Back button to return to the list of CRM Links connected to this region. Repeat steps 4 and 5 until all CRM Links for this region are stopped.

**Note:** Once the region is back in service you can repeat this process to restart the CRM Links

# Shutting Down a Mainframe

If you must shut down the mainframe on which the CRM is running, you should stop the CRM prior to shutting down the mainframe. To shut down the CRM, see [“Stopping the CRM.”](#) Once the CRM is down, all Gateways connected to this CRM will automatically terminate.

# Shutting Down an Instance of WebLogic Server

If you need to shut down an instance of WebLogic Server, and it has a Gateway running in it, you must stop the Gateway prior to shutting down the WebLogic Server instance. To shut down a Gateway, see [“Starting and Stopping a Gateway.”](#) When the instance of WebLogic Server is restarted, the Gateway will automatically start during WebLogic Server startup as long as it is deployed.

# Unplanned Outages

When you plan for outages and perform the appropriate steps to administer your WebLogic JAM components, you can institute predictable and continuous support for your server environment. Occasionally components in your system will terminate unexpectedly.

The following are some unexpected outages that you might plan for.

- [Application Region Terminates](#)
- [z/OS or OS/390 Region Terminates](#)



## Application Region Terminates

The JAM gateway and CRM cannot detect when an application region (CICS, IMS or batch) terminates unexpectedly. There is no notification of this termination to the CRM, and the status of the link remains active. The WebLogic JAM Gateway and CRM continue to process requests on that link. When a request is sent to the inactive region over the active link, then the request will fail and a failure message is returned to the CRM and the Gateway.

As soon as the region terminates, the administrator should use the CRM console or command line administration tool to terminate the link to the region. When the region is restarted, use the tools to restart the link to the region. For information on how to restart the link, see [“Starting and Stopping CRM Links.”](#)

If transactions were in flight when the region terminated, then you should check to see if any transactions must be recovered. For more information on recovering transactions, see [“Recovery.”](#)

## z/OS or OS/390 Region Terminates

If the mainframe region where the CRM process is running terminates unexpectedly, a WebLogic JAM Gateway associated with the CRM recognizes that the CRM is no longer available and shuts down. However, the Gateway will go into an automatic restart state. When the CRM is restarted, all of the WebLogic JAM Gateways associated with the CRM will automatically reestablish communication with the CRM.

The CRM should only be restarted after all of its application regions are ready for use. If the CRM cannot automatically connect to the region, you should use the command line or console tools to start the CRM links when the region is ready. For more information, see [“Starting and Stopping CRM Links.”](#)

If transactions were in flight when the mainframe region terminated, then you should check to see if any transactions must be recovered. For more information on recovering transactions, see [“Recovery.”](#)



# A Mainframe Connectivity Worksheet

Detailed information and steps for obtaining WebLogic JAM connectivity parameters and entering them into the WebLogic JAM extension of the WebLogic Administration Console are available in [“Configuring WebLogic JAM Connectivity,”](#) in the *BEA Java Adapter for Mainframe Configuration and Administration Guide*.

## CRM Definitions

For information about obtaining the CRM definition parameters, see [“Step 2: Define Where the CRM Will Run.”](#)

WebLogic Administration Console Parameter Field Name	Your Environment Values	Associated VTAM Definition (if applicable)
Name		
Listen Address		
Listen Port		
Logical Unit		ACBNAME

WebLogic Administration Console Parameter Field Name	Your Environment Values	Associated VTAM Definition (if applicable)
Stack Type		VTM28

## Regions

For information about obtaining these parameters, see “[Step 4: Connect the CRM to Back-End Systems on the Mainframe](#)” in the *BEA Java Adapter for Mainframe Configuration and Administration Guide*.

---

## CICS Regions

WebLogic Administration Console Parameter Field Name	Your Environment Values	Associated VTAM Definition (if applicable)
Name		
Logical Unit		ACBNAME(VTAM APPLID)

## IMS Regions

WebLogic Administration Console Parameter Field Name	Your Environment Values	Associated APPC/MVS LU Definition (if applicable)
Name		
Logical Unit		LUADD ACPNAME)

## Batch Regions

WebLogic Administration Console Parameter Field Name	Your Environment Values	Associated APPC/MVS LU Definition (if applicable)
Name		
Logical Unit		LUADD ACPNAME

## CRM Links

For information about obtaining these parameters, see and [“Determining Connection Characteristics”](#) in the *BEA Java Adapter for Mainframe Configuration and Administration Guide*.

## CICS

WebLogic Administration Console Parameter Field Name	Your Environment Values	Associated CICS Session Definition (if applicable)
Name		
Region Name		Refer to Name value from <a href="#">“CICS Regions.”</a>
Logmode Name		MODENAME
Maximum Session		MAXIMUM(SESSNBR)
Minimum Winner Session		MAXIMUM(SESSNBR-WINNER)
Security Level		ATTACHSEC(LOCAL IDENTIFY VERIFY)
Default User ID		
<b>Note:</b> Only valid if Security Level is Identify		

---

# IMS

WebLogic Administration Console Parameter Field Name	Your Environment Values	Associated VTAM Definition (if applicable)
Name		
Region Name		Refer to Name value from “IMS Regions.”
Logmode Name		DLOGMOD
Maximum Session		DSESLIM
Minimum Winner Session		DMINWNL
Security Level Local, Identify, Verify)		SECACPT(NONE   ALREADYV   CONV)
Default User ID		
<b>Note:</b> Only valid if Security Level is Identify		

**Batch**

<b>WebLogic Administration Console Parameter Field Name</b>	<b>Your Environment Values</b>	<b>Associated VTAM Definition (if applicable)</b>
Name		
Region Name		Refer to Name value from <a href="#">“Batch Regions.”</a>
Logmode Name		DLOGMOD
Maximum Session		DSESLIM
Minimum Winner Session		DMINWNL
Security Level Local, Identify, Verify)		SECACPT(NONE   ALREADYV   CONV)
Default User ID		



---

# Index

## A

- administration
  - CRM 1-9
  - transactions 1-9
  - WebLogic Administration Console 1-9
- affinity
  - transactional 4-12
- APPC
  - batch connection 2-31
  - batch region definition 2-32
  - CICS connection protocol 2-18
  - CICS region definition 2-18
  - IMS connection protocol 2-25
  - IMS region definition 2-26
  - logical unit 2-12
  - programs invoke J2EE applications 3-28
  - service configuration 3-12, 3-19
  - services
    - delete 5-47
    - edit 5-44
    - enabling 5-46, 6-46
  - traces 6-13, 7-2
  - transaction program
    - data translation requirements 3-19
  - transaction program access 3-19
  - transaction programs 3-18
- application integration
  - access WebLogic applications 3-3
  - considerations 3-2
  - expose mainframe application 3-3
- APPLID

- VTAM 2-12

- transactional support 8-2

- architecture 1-2

## B

- batch environment
  - transactional support 8-3
- batch region
  - adding APPC 2-32
  - connecting to 2-30
  - create 2-35
  - defining logical unit 2-31
  - logical unit 2-30
  - maximum sessions 2-31
  - parameter definitions 2-33
  - scheduler 2-32
  - security 2-31
  - sessions 2-31
  - VTAM logmode 2-31
- blob log files 8-4

## C

- CICS
  - connecting to 2-17
  - DPL program
    - accessing J2EE applications 3-26
  - DPL programs 3-4
    - transaction ID 3-5
  - program resource definition for DPL
    - program 3-5

- 
- region
    - adding APPC 2-18
    - configuration steps 2-17
    - connection definition 2-18
    - create 2-35
    - installing connection and session definition 2-20
    - logical unit 2-18
    - maximum sessions 2-19
    - parameter definitions 2-21, 2-22
    - security 2-18
    - session definition 2-19
    - sessions 2-19
    - verify connection and session definition 2-20
    - VTAM logmode 2-19
    - transactional support 8-3
  - clusters
    - multi-Gateway 4-8
    - supported features 4-7
  - cold start 5-6
    - transactional links 8-8
      - steps for 8-9
  - COMMAREA
    - data formats for DPL program 3-4
  - Communication Resource Manager, see CRM
  - configuration
    - application integration 1-4
    - batch region
      - adding APPC 2-32
      - CRM link 2-33
      - logical unit 2-30, 2-33
    - CICS regions
      - adding APPC 2-18
      - connecting to 2-17
      - connection definition 2-18
      - CRM link 2-22
      - logical unit 2-18, 2-21
      - session definition 2-19
    - clustered environment 4-8, 4-9
    - connectivity 1-4, 2-2
      - verify 2-56
  - CRM
    - definition parameters 2-10
    - logical unit 2-13
    - on the mainframe 2-10
  - determining connections 2-14
  - Gateways
    - first instance 4-5
    - multiple 4-4
    - subsequent instance 4-5
  - general steps 1-10
  - IMS regions
    - adding APPC 2-26
    - connecting to 2-24
    - CRM link 2-28
    - logical unit 2-24, 2-27
  - mainframe 2-3
  - mainframe clients
    - accessing J2EE applications 3-28
  - mainframe clients invoke EJBs 3-30
  - mainframe clients invoke J2EE applications 3-25, 3-28
  - mainframe clients invoke JMS Events 3-35
  - overview 1-4
  - security levels 2-16
    - batch region 2-31
    - CICS region 2-18
    - IMS region 2-25
  - services
    - accessing J2EE applications 3-26
    - APPC 3-12, 3-19
    - DPL 3-5
  - support for clusters 1-6
  - SYNCLVL support 8-2
  - transactional support 8-2
    - batch environment 8-3
    - CICS environment 8-3
    - IMS environment 8-3
- connectivity

---

- batch region 2-31
- CICS connection definition 2-18
- CICS region
  - automatic session initialization 2-19
  - install connection and session definition 2-20
  - verify connection and session definition 2-20
- CICS session definition 2-19
- CRM
  - parameters 2-10
  - to mainframe applications 2-14
- determine connections 2-14
- general description 2-2
- general steps to establish 2-3
- logmode 2-15
- mainframe configuration 2-3
- protocol
  - CICS region 2-18
  - IMS region 2-25
- session considerations 2-15
- to batch regions 2-30
- to CICS regions 2-17
- to IMS regions 2-23
- verify configuration 2-56
- worksheet A-1

console commands

- CRM 7-19

CPI-C

- exposing program as APPC service 3-18

CRM

- architecture 1-2
- batch region
  - parameter definitions 2-33
- CICS region
  - parameter definitions 2-21
- configuring logical unit 2-13
- connecting to mainframe applications 2-14
- connectivity 2-2
- console commands 7-19
- create definition 2-41
- defining logical unit 2-11
- definition
  - delete 5-14
  - edit 5-12
  - list 5-10
- determine connections 2-14
- establishing links 1-3
- Gateway definition 2-51
- IMS region
  - parameter definitions 2-27
- link 2-14
  - activating 7-12
    - MVS 7-13
    - z/os or OS/390 Unix 7-15
  - cold start 5-6
  - create 2-46
  - de-activating 7-15
    - MVS 7-16
    - z/os or OS/390 Unix 7-18
  - delete 5-25
  - deploying 5-24
  - edit 5-21
  - list 5-18
  - logmode 2-15
  - multiple 4-2
  - parameter definitions 2-22, 2-28, 2-33
  - planned shut down 9-2
  - security levels 2-16
    - batch region 2-31
    - CICS region 2-18
    - IMS region 2-25
  - sessions 2-15
  - start 6-23
  - status 6-17
  - stop 5-16, 6-23
  - transactional support 8-2
    - batch environment 8-3
    - CICS environment 8-3
    - IMS environment 8-3

---

- recovery 8-10, 8-11
- restarting 8-8
- load balancing
  - mainframe to WebLogic 4-7
- location of CRM executable 7-5
- location of crmdown executable 7-9
- location of crmlkoff executable 7-16
- location of crmlkon executable 7-13
- location of CRMLOGS executable 8-6
- logical unit 2-12
  - CICS region 2-18
- logical unit parameter definitions 2-12
- multiple Gateways 2-52, 4-2
- MVS environment
  - starting the CRM
- OS/390 Unix environment
  - starting the CRM 7-7
- overview 1-3
- parameters for configuring connectivity 2-10
- performance option 7-3
- region size
  - crmdown command 7-9
  - crmlkoff command 7-17
  - crmlkon command 7-13
  - CRMLOGS command 8-6
  - crmstart command 7-5
- setting C/C++ runtime library
  - crmdown command 7-9
  - crmlkoff command 7-17
  - crmlkon command 7-13
  - CRMLOGS command 8-6
  - crmstart command 7-5
- setting CRM command parameters 7-5
- setting crmdown command parameters 7-9
- setting crmlkoff command parameters 7-16
- setting crmlkon command parameters 7-13
- setting CRMLOGS command

- parameters 8-5
- setting ENVFILE DD name
  - crmdown command 7-9
  - crmlkoff command 7-17
  - crmlkon command 7-13
  - CRMLOGS command 8-6
  - crmstart command 7-5
- setting LE runtime library
  - crmdown command 7-9
  - crmlkoff command 7-17
  - crmlkon command 7-13
  - CRMLOGS command 8-6
  - crmstart command 7-5
- setting LE runtime operands
  - crmdown command 7-9
  - crmlkoff command 7-16
  - crmlkon command 7-13
  - CRMLOGS command 8-6
  - crmstart command 7-5
- shut down
  - z/os or OS/390 Unix 7-8
- single point of failure 4-4
- starting
  - CRM command 7-1
- status 6-2
- traces 6-6, 7-2
- transaction log files 8-4
  - blob 8-4
  - restart 8-4
  - viewing 8-5
- transactional resources 1-3
- where to run 2-10
- z/OS environment
  - starting the CRM 7-7
- CRM command
  - APPC API trace settings 7-2
  - starting CRM 7-1
  - starting the CRM 7-7
  - trace settings 7-2
- crmdown command
  - shutting down the CRM 7-8

---

crmlkoff command  
    de-activating CRM links 7-15, 7-16, 7-18  
crmlkon command  
    activating CRM links 7-12, 7-13, 7-15

## D

data integrity 8-2  
    transaction log files 8-3  
data translation  
    APPC transaction program 3-19  
    considerations 3-2  
    DPL program 3-4  
    IMS transaction program 3-11  
DataViews  
    data formats for APPC transaction program 3-19  
deploying CRM links 5-24  
diagnostics  
    APPC API traces 6-13, 7-2  
    CRM traces 6-6, 7-2  
    Gateway traces 6-35  
disconnect  
    Gateways 4-6  
Distributed Program Link, see DPL  
DL/I  
    access J2EE applications 3-28  
    exposing program as APPC service 3-11  
DPL  
    access J2EE applications 3-26  
    data translation requirements 3-4  
    exposing program as a service 3-4  
    program access 3-5  
    services  
        delete 5-41  
        edit 5-38  
        enabling 5-40, 6-46

## E

EJB  
    applications invoked by mainframe 3-30  
    export definition  
        delete 5-53  
        edit 5-50  
        enabling local service 5-52, 6-49, 6-51  
Enterprise Java Bean, see EJB  
ENV 7-5  
ENVFILE  
    setting  
        crmdown command 7-9  
        crmlkoff command 7-17  
        crmlkon command 7-13  
        CRMLOGS command 8-6  
        crmstart command 7-5  
environment file  
    set command  
        de-activating CRM links 7-17  
        starting CRM 7-5  
        starting CRM links 7-13  
        stopping CRM 7-9  
        viewing CRM logs 8-6  
environment variables  
    setting in MVS 7-4  
    setting in z/OS or OS/390 Unix 7-7  
export  
    EJBs to mainframe application 3-30  
export definition 3-3

## F

failover 2-52, 4-2  
    processing  
        mainframe to WebLogic 4-11  
        WebLogic to mainframe 4-11  
    support for 4-11

---

## G

### Gateway

- architecture 1-2
- connecting to CRM 2-51
- connection
  - first 4-5
  - multiple 4-5
- connectivity 2-2
- define 2-51
- delete 5-34
- edit 5-31
- failover 2-52
- list 5-29, 6-29
- load balancing 2-52
- monitor 6-31
- multiple 2-52, 4-2
- multiple instances 1-3
- overview 1-3
- start 6-41
- statistics 6-34
- stop 6-41
- traces 6-35

### Gateways

- clustered environment
  - multiple CRM 4-9
  - single CRM 4-8
- disconnect 4-6
- shut down 4-6

## I

### IMS

- accessing J2EE applications 3-28
- connecting to 2-23
- region
  - adding APPC 2-26
  - configuration steps 2-24
  - create 2-35
  - defining logical unit 2-25
  - logical unit 2-24
  - maximum sessions 2-25

- parameter definitions 2-27, 2-28
- scheduler 2-26
- security 2-25
- sessions 2-25
- VTAM logmode 2-25

- transaction program
  - data translation requirements 3-11
- transaction program access 3-12
- transaction programs 3-11
- transactional support 8-3

Input/Output Program Control Block, see  
IOPCB

### IOPCB

- data formats for IMS transaction  
program 3-11

## J

### J2EE

- applications
  - CICS program access 3-26
  - IMS program access 3-28
- applications invoked by mainframe  
clients 3-25, 3-28
- invoking applications 3-25

jam.ear file 2-6

jam.jar file 2-5

Java Transactional API, see JTA

### JCL

- CRM shutdown 7-9
- set commands
  - C/C++ runtime library
    - crmdown command 7-9
    - crmlkoff command 7-17
    - crmlkon command 7-13
    - CRMLOGS command 8-6
    - crmstart command 7-5
  - CRM executable location 7-5
  - crmdown executable location 7-9
  - crmlkoff executable location 7-16
  - crmlkon executable location 7-13

---

CRMLOGS executable location 8-6  
environment file

- crmdown command 7-9
- crmlkoff command 7-17
- crmlkon command 7-13
- CRMLOGS command 8-6
- crmstart command 7-5

LE runtime library

- crmdown command 7-9
- crmlkoff command 7-17
- crmlkon command 7-13
- CRMLOGS command 8-6
- crmstart command 7-5

LE runtime operands

- crmdown command 7-9
- crmlkoff command 7-16
- crmlkon command 7-13
- CRMLOGS command 8-6
- crmstart command 7-5

region size

- crmdown command 7-9
- crmlkoff command 7-17
- crmlkon command 7-13
- CRMLOGS command 8-6
- crmstart command 7-5

set CRM command parameters 7-5

set crmdown command parameters  
7-9

set crmlkoff command parameters  
7-16

set crmlkon command parameters 7-  
13

set CRMLOGS command  
parameters 8-5

sets ENVFILE DD name

- crmdown command 7-9
- crmlkoff command 7-17
- crmlkon command 7-13
- CRMLOGS command 8-6
- crmstart command 7-5

starting the CRM 7-4

stopping the CRM 7-9

viewing CRM log files 8-5

starting the CRM 7-4

JMS

applications invoked by mainframe 3-35

Event

- delete 5-59

- disabling local service 6-51

- edit 5-56

- enabling local service 5-58, 6-49

job

MVS 2-10

job control language, see JCL

JTA

transaction log files 8-4

transactional support 1-9

## L

Language Environment

setting runtime library

- crmdown command 7-9

- crmlkoff command 7-17

- crmlkon command 7-13

- CRMLOGS command 8-6

- crmstart command 7-5

setting runtime operands

- crmdown command 7-9

- crmlkoff command 7-16

- crmlkon command 7-13

- CRMLOGS command 8-6

- crmstart command 7-5

links

CRM 1-3

- start 6-23

- status 6-17

- stop 6-23

load balancing 2-52

mainframe to WebLogic 4-7

multiple regions 4-2

Weblogic to mainframe 4-10

---

- log files
  - blob 8-4
  - CRM 8-4
    - viewing 8-5
  - CRM restart 8-4
  - JTA 8-4
  - mainframe applications 8-8
  - transaction 8-3
- logical unit
  - APPC 2-12
  - batch region 2-31, 2-32
  - batch region definition 2-30
  - CICS region definition 2-18
  - CRM and VTAM APPLID 2-12
  - CRM definition 2-11
  - CRM parameters 2-12
  - editing 5-4
  - IMS region 2-25, 2-26
  - IMS region definition 2-24
  - scheduler
    - batch region 2-32
    - IMS region 2-26
  - SNA network 2-2
  - transactional support 8-2
- logmode
  - connection considerations 2-15
  - VTAM
    - sessions for batch region 2-31
    - sessions for CICS region 2-19
    - sessions for IMS region 2-25
- LU 6.2 2-2

## M

- mainframe
  - application log files 8-8
  - applications
    - invoking EJB 3-30
    - invoking JMS Event 3-35
  - client applications 3-25
    - APPC programs 3-28

- connecting CRM to access applications 2-14
  - connectivity 2-3
  - planned shut down 9-6
  - running the CRM 2-10
- mainframe shut down
  - mainframe 9-6
- maximum sessions
  - batch regions 2-31
  - CICS regions 2-19
  - considerations 2-15
  - IMS regions 2-25
- minimum winner sessions 2-15
  - batch region 2-31
  - CICS region 2-19
  - IMS region 2-25
- MVS
  - activating links 7-13
  - CRM log files
    - viewing 8-5
  - de-activating links 7-16
  - running the CRM 2-10
  - starting the CRM 7-4

## N

- network connections 2-3

## O

- OS/390
  - CRM log files
    - viewing 8-7
- OS/390 Unix
  - activating links 7-15
  - CRM shutdown 7-8
  - de-activating links 7-18
  - running the CRM 2-10
  - starting the CRM 7-7
- os/390 Unix
  - setting environment variables 7-7



---

overview 1-1

## P

performance

CRM option 7-3

protocol

APPC 1-3

CICS connection protocol 2-18

IMS connection protocol 2-25

TCP/IP 1-3

two-phase commit 8-2

blob log file 8-4

## Q

queue

JMS 3-35

## R

recovery 8-10

initiating 8-11

regions

batch

adding APPC 2-32

connecting to 2-30

logical unit 2-30, 2-31

CICS

adding APPC 2-18

configuration steps 2-17

connecting to 2-17

connection definition 2-18

logical unit 2-18

session definition 2-19

creating 2-35

delete 5-7

IMS

adding APPC 2-26

configuration steps 2-24

connecting to 2-23

logical unit 2-24, 2-25

listing 5-2

modifying 5-2

MVS 2-10

remote

shutting down 9-2

reliability 4-1

RRS

transaction manager 8-3

runtime library

C/C++

crmdown command 7-9

crmlkoff command 7-17

crmlkon command 7-13

CRMLOGS command 8-6

crmstart command 7-5

LE

crmdown command 7-9

crmlkoff command 7-17

crmlkon command 7-13

CRMLOGS command 8-6

crmstart command 7-5

## S

scalability 4-1

scheduler

batch region 2-32

IMS region 2-26

security

considerations 2-16

with batch region 2-31

with CICS region 2-18

with IMS region 2-25

service

APPC

delete 5-47

edit 5-44

enabling 5-46

parameters 3-12, 3-19

definition 3-3

---

- disable
  - exported EJB 6-51
  - JMS Event 6-51
  - WebLogic Integration Event 6-51
- DPL
  - delete 5-41
  - edit 5-38
  - enabling 5-40
  - parameters 3-5
- enable
  - exported EJB 5-52, 6-49
  - JMS Event 5-58, 6-49
  - WebLogic Integration Event 5-64, 6-49
- enabling
  - APPC 6-46
  - DPL 6-46
- services
  - exposing J2EE applications 3-25
- sessions
  - batch region
    - VTAM logmode 2-31
- CICS region
  - automatic initialization 2-19
  - install definition 2-20
  - verify definition 2-20
  - VTAM logmode 2-19
- configuration overview 2-3
- connection considerations 2-15
- contention
  - batch region 2-31
  - CICS region 2-19
  - considerations 2-15
  - IMS region 2-25
- IMS region
  - VTAM logmode 2-25
- maximum
  - batch regions 2-31
  - CICS regions 2-19
  - description 2-15
  - IMS regions 2-25
- minimum winner 2-15
  - batch region 2-31
  - CICS region 2-19
  - IMS region 2-25
- multiple session support 2-13
  - batch region 2-31
  - IMS region 2-25
- pools 2-15
- set commands
  - C/C++ runtime library
    - crmdown command 7-9
    - crmlkoff command 7-17
    - crmlkon command 7-13
    - CRMLOGS command 8-6
    - crmstart command 7-5
  - CRM executable location 7-5
  - crmdown executable location 7-9
  - crmlkoff executable location 7-16
  - crmlkon executable location 7-13
  - CRMLOGS executable location 8-6
  - environment file
    - crmdown command 7-9
    - crmlkoff command 7-17
    - crmlkon command 7-13
    - CRMLOGS command 8-6
    - crmstart command 7-5
  - JCL
    - viewing CRM log files 8-5
  - JCLstarting CRM 7-4
  - JCLstopping CRM 7-9
  - LE runtime library
    - crmdown command 7-9
    - crmlkoff command 7-17
    - crmlkon command 7-13
    - CRMLOGS command 8-6
    - crmstart command 7-5
  - LE runtime operands
    - crmdown command 7-9
    - crmlkoff command 7-16
    - crmlkon command 7-13
    - CRMLOGS command 8-6

---

- crmstart command 7-5
- sets CRM command parameters 7-5
- sets crmdown command parameters 7-9
- sets crmlkoff command parameters 7-16
- sets crmlkon command parameters 7-13
- sets CRMLOGS command parameters 8-5
- sets ENVFILE DD name
  - crmdown command 7-9
  - crmlkoff command 7-17
  - crmlkon command 7-13
  - CRMLOGS command 8-6
  - crmstart command 7-5
- sets region size
  - crmdown command 7-9
  - crmlkoff command 7-17
  - crmlkon command 7-13
  - CRMLOGS command 8-6
  - crmstart command 7-5
- shut down
  - planned outages 9-1
  - remote regions 9-2
  - WebLogic Server 9-6
- shutdown process
  - Gateways 4-6
- SNA 2-2
- statistics
  - Gateway 6-34
- system outages
  - planned 9-1, 9-6
  - WebLogic Server 9-6

## T

- traces
  - APPC 6-13, 7-2
  - CRM 6-6
  - CRM command 7-2
  - Gateway 6-35
- transaction ID
  - APPC transaction program 3-19

- DPL program 3-5
- IMS transaction program 3-12
- transactions
  - affinity 4-12
  - batch environment 8-3
  - CICS environment 8-3
  - coordinating resources 1-3
  - distributed 4-12
  - IMS environment 8-3
  - link
    - restarting
      - cold start 8-8
      - warm start 8-8
  - link support for 8-2
  - log files
    - blob 8-4
    - CRM 8-4
      - viewing 8-5
    - data integrity 8-3
    - JTA 8-4
    - mainframe applications 8-8
    - restart 8-4
  - loosely-coupled 4-12
  - manager 8-1
    - RRS 8-3
  - recovery 8-10
    - initiating 8-11
  - register Gateways
    - Gateway
      - register with transaction manager 8-1
  - SYNCLVL support
    - batch region 2-31
    - CRM logical unit 2-12
    - IMS region 2-25
  - tightly-coupled 4-12
  - two-phase commit protocol 8-2
  - VTAM APPLID definition 8-2
- two-phase commit protocol
  - blob log file 8-4
  - coordinating transactions 8-2

---

## V

### VTAM

- accessing CICS region 2-18
- administrator 2-11
- APPLID 2-12
  - SYNCLVL support 8-2
- batch region
  - defining logical unit 2-32
  - logmode name for sessions 2-31
- CICS region
  - creating CRM logical unit 2-18
  - logmode name for sessions 2-19
- creating logical unit for CRM 2-11
- IMS region
  - defining logical unit 2-26
  - logmode name for sessions 2-25

## W

### warm start

- transactional links 8-8, 8-10

### WebLogic Administration Console

- accessing WebLogic JAM 1-6
- APPC API trace settings 6-13
- batch region
  - parameter definitions 2-33
- CICS region
  - parameter definitions 2-21, 2-22
- create
  - CRM definition 2-41
  - CRM Link 2-46
  - region definitions 2-35

### CRM

#### link

- start 6-23
- status 6-17
- stop 6-23

### CRM definition parameters 2-10

### CRM logical unit 2-13

### CRM status

- monitor 6-2

### CRM trace settings 6-6

### define Gateway 2-51

### EJBs

- exposing to the mainframe 3-30

### Gateway

- list 6-29
- monitor 6-31
- start 6-41
- statistics 6-34
- stop 6-41

### Gateway trace settings 6-35

### IMS region

- parameter definitions 2-27, 2-28

### JMS Event

- exposing to the mainframe 3-35

### overview 1-1

### services

- APPC 3-12, 3-19
- DPL 3-5

### WebLogic Integration

#### Event

- delete 5-64
- disabling local service 6-51
- edit 5-62
- enabling local service 5-64, 6-49

### WebLogic JAM

- home page 1-7

### WebLogic Server

- planned shut down 9-6

### worksheet

- connectivity A-1

## Z

### z/OS

### activating links 7-15

### CRM log files

- viewing 8-7

### CRM shutdown 7-8

### de-activating links 7-18

### running the CRM 2-10

---

setting environment variables 7-7  
starting the CRM 7-7