ClaimCenter Installation Guide

Release 6.0.8

Product Name: Guidewire ClaimCenter
Product Release: 6.0.8
Document Name: ClaimCenter Installation Guide
Document Revision: 05-February-2013

Contents

About This Document

This document describes the process to install and deploy ClaimCenter.

This topic includes:

Intended Audience

This document is intended for the following readers:

- System administrators
- Developers who need to install the ClaimCenter server and associated administrative tools on a server and one or more workstations
- Configuration managers responsible for determining which configuration files need to be placed into source control systems
- Build masters charged with packaging configured applications for testing and production

Assumed Knowledge

This document assumes that you are already familiar with the following topics:

- Installing systems with your operating system
- Installing and administering the database of your choice
- Installing and administering the application server of your choice

Related Documents

See the following Guidewire documents for further information:

ClaimCenter System Administration Guide – Provides guidance for the ongoing management of a ClaimCenter system. This document is intended to help system administrators monitor ClaimCenter, manage its security, and take care of routine tasks such as system backups, logging, and importing files.

ClaimCenter Upgrade Guide – Provides instructions to upgrade ClaimCenter.

ClaimCenter Reporting Guide – This document contains an overview of ClaimCenter reports. ClaimCenter uses InetSoft Style Report Enterprise Edition to provide a number of ClaimCenter-specific reports. This document describes how to install and configure the InetSoft application to work with ClaimCenter reports, as well as how to create and run reports.

Conventions In This Document

Text style	Meaning	Examples
italic	Emphasis, special terminology, or a book title.	A *destination* sends messages to an external system.
bold	Strong emphasis within standard text or table text.	You **must** define this property.
narrow bold	The name of a user interface element, such as a button name, a menu item name, or a tab name.	Next, click **Submit**.
`monospaced`	Literal text that you can type into code, computer output, class names, URLs, code examples, parameter names, string literals, and other objects that might appear in programming code.	Get the field from the `Address` object.
`monospaced italic`	Parameter names or other variable placeholder text within URLs or other code snippets.	Use `getName(`*`first`*`, `*`last`*`)`. `http://`*`SERVERNAME`*`/a.html`.

Support

For assistance with this software release, contact Guidewire Customer Support:

- At the Guidewire Resource Center – `http://guidewire.custhelp.com`
- By email – `support@guidewire.com`
- By phone – +1-650-356-4955

chapter 1

Introduction to Installation

This guide describes how to install ClaimCenter.

This topic describes the installation options that are available and provides guidance on selecting which option to choose.

The *ClaimCenter Installation Guide* includes:

- An overview that discusses the different ways to install ClaimCenter that starts with this topic.
- “Preparing a ClaimCenter Environment” on page 11 discusses supported database and application server platforms and provides steps to prepare a development or production environment for ClaimCenter.
- “Installing a ClaimCenter Development Environment” on page 31 explains how to install a ClaimCenter development environment using the QuickStart server and database or Tomcat.
- “Installing a ClaimCenter Production Environment” on page 41 explains how to deploy ClaimCenter to an application server and database server production environment.
- “Commands Reference” on page 69 lists and describes the QuickStart and build commands. Other ClaimCenter command line utilities are described in “ClaimCenter Administrative Commands” on page 169 in the *System Administration Guide*.

This topic includes:

- “Selecting an Installation Scenario” on page 8
- “Viewing the Development and Production Environments” on page 10

Selecting an Installation Scenario

Choose an installation scenario based on the role of the person who uses the ClaimCenter installation. The following table lists installation scenarios that Guidewire recommends for each role.

Role	Description	Consult this Section
Demonstrator or Trainer	Starts up the application quickly, loads sample data and demonstrates features.	"Installation Environments Overview" on page 12 "Installing a ClaimCenter Development Environment" on page 31
Application Developer	Changes the behavior of the application including the user interface, rules, and application logic.	"Installation Environments Overview" on page 12 "Installing a ClaimCenter Development Environment" on page 31
Integration Developer	Develops software to connect ClaimCenter to external systems.	"Installation Environments Overview" on page 12 "Installing a ClaimCenter Development Environment" on page 31
Conversion Developer	Performs analysis and mapping of legacy data structures to Guidewire application data model.	"Installation Environments Overview" on page 12 "Installing a ClaimCenter Development Environment" on page 31
Build master deploying to testing and production	Deploys finished application to test and production environments.	"Installation Environments Overview" on page 12 "Installing a ClaimCenter Production Environment" on page 41.

List of Installation Options

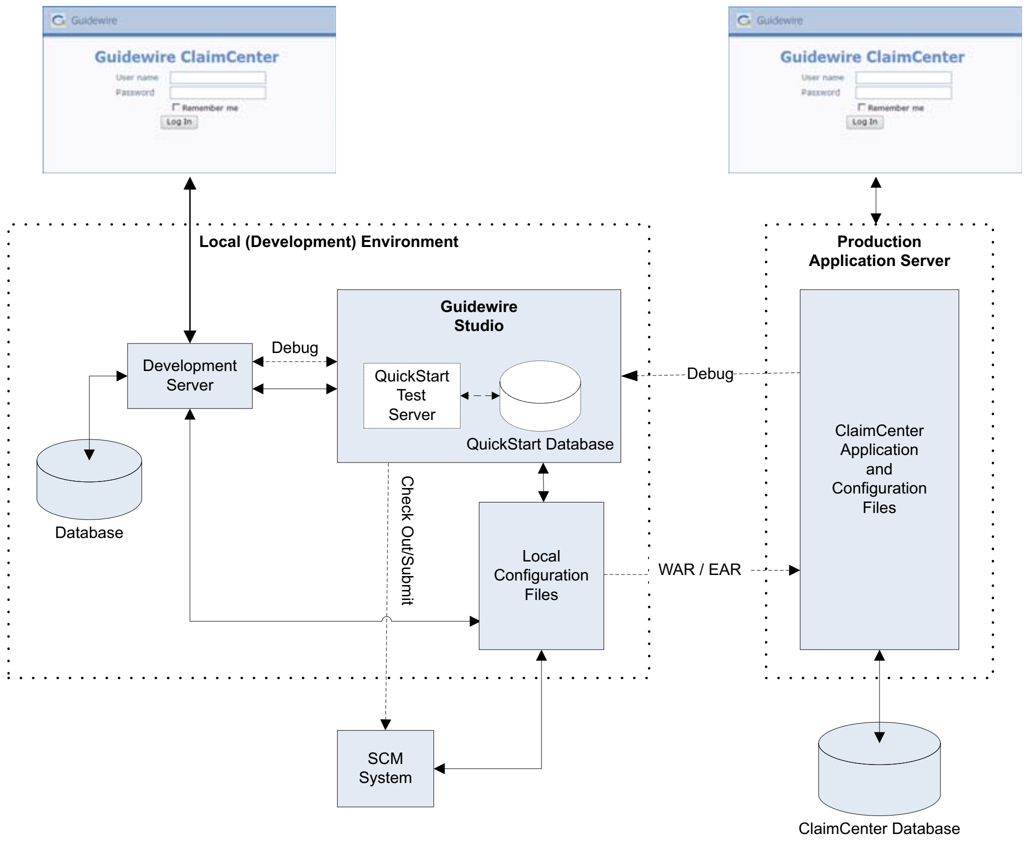
If you are still questioning your installation options based on your role, this section can provide additional clarification.

Install element	Bundled or Optional	Good to know	Links for more Information
QuickStart Application Server	Bundled	You can immediately use the QuickStart server without configuring it. It does not build a WAR or EAR file which is necessary before deployment with other servers. This can be used for demonstration or development environments. Guidewire does not support QuickStart for a production environment. ClaimCenter can be quickly started in development (dev) mode using the steps in "Installing the QuickStart Development Environment" on page 32. ClaimCenter can not run in production (prod) mode on QuickStart. For more information on server modes, see "Determining Server Mode" on page 62.	"Advantages to Using the QuickStart Software" on page 32 "QuickStart Application Server" on page 34

Install element	Bundled or Optional	Good to know	Links for more Information
JBoss Application Server	Optional	Suitable for production environments.	"Configuring the Application Server" on page 13 "Installing a JBoss Production Environment" on page 56
Tomcat Application Server	Optional	Suitable for production environments. You can use Tomcat instead of the QuickStart server for development work although it requires additional configuration.	"Configuring the Application Server" on page 13 "Installing a Tomcat Development Environment" on page 35 "Installing a Tomcat Production Environment" on page 57
WebSphere Application Server	Optional	Suitable for production environments.	"Configuring the Application Server" on page 13 "Installing a WebSphere Production Environment" on page 59
WebLogic Application Server	Optional	Suitable for production environments.	"Configuring the Application Server" on page 13 "Installing a WebLogic Production Environment" on page 58
QuickStart Database	Bundled	You can immediately use the QuickStart database in file mode. ClaimCenter creates and stores the database files within the `tmp` directory of the local drive. You can customize this location. You can use the QuickStart database for demonstration or development environments. Guidewire does not support QuickStart for a production environment. Guidewire does not support upgrades to the QuickStart database. Configuring your application sometimes requires extending the data model, which might require dropping the database. You can not have more than one connection at a time.	"Advantages to Using the QuickStart Software" on page 32 "Using the QuickStart Database" on page 37
Oracle Database	Optional	You can use this database for development and production.	"Configuring the Database" on page 20 "Configuring Oracle for ClaimCenter" on page 21 "Configuring a Database Connection" on page 44
SQL Server Database	Optional	You can use this database for development and production.	"Configuring the Database" on page 20 "Configuring SQL Server for ClaimCenter" on page 23 "Configuring a Database Connection" on page 44

Viewing the Development and Production Environments

The following diagram illustrates how the ClaimCenter development and production environment interact.



Dotted lines indicate actions that you perform. For example, you create a WAR or EAR file from your configured development environment and move it to the production server.

To assist with this development and testing process, Guidewire bundles the following with the ClaimCenter application:

- A QuickStart development server
- A QuickStart database
- A QuickStart test server that you cannot control
- A QuickStart test database that is separate from the QuickStart database

Guidewire bundles the QuickStart test server and test database for testing. These components are internally controlled. You can use either the bundled QuickStart development server bundled with ClaimCenter or use an external application server such as Tomcat. If you use the QuickStart method, then the default development server is Jetty and the database is H2. Guidewire does not support the QuickStart application server or database for a production environment.

chapter 2

Preparing a ClaimCenter Environment

This topic describes how to install and configure necessary system components so that your network can support ClaimCenter. It also includes preparatory steps for deploying a production instance of ClaimCenter.

The versions of third-party products that Guidewire supports for this release are subject to change without notice. See the *Guidewire Platform Support Matrix* for current system and patch level requirements. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.

If you are upgrading an existing ClaimCenter installation, see the *ClaimCenter Upgrade Guide*.

If you use ClaimCenter with Standard Reporting, you must do additional configuration. See the *ClaimCenter Reporting Guide* for details.

This topic includes:

Installation Environments Overview

ClaimCenter has a typical J2EE three-tier architecture: client, application server, and database server. For the application to function, install and configure each tier correctly. Before you get started, have the appropriate software versions to support ClaimCenter. Guidewire strongly recommends that you obtain support contracts with vendors for all tiers of your application infrastructure.

Although production environments can run on operating systems other than Windows, you must build ClaimCenter on a Windows system prior to deploying ClaimCenter. Development environments must be on Windows. See “Development Workstation Information” on page 27.

Production Environments

Guidewire strongly recommends that you allocate dedicated hardware for the application server and database server tiers. Reserve this hardware solely for ClaimCenter. Using dedicated hardware is best for performance and for isolating the cause of any issues that arise.

Although production environments can run on operating systems other than Windows, you must build ClaimCenter on a Windows system prior to deploying ClaimCenter.

ClaimCenter requires a 64-bit operating system and JVM for a production installation.

To install a production environment, first review all of the information in “Preparing a ClaimCenter Environment” on page 11. Then proceed to “Installing a ClaimCenter Production Environment” on page 41.

Development Environments

Guidewire only supports the use of the bundled QuickStart application server or Tomcat for development environments.

For all development environments, review the following topics:

- “Development Workstation Information” on page 27
- “Client Information” on page 27
- “Creating Accounts to Run ClaimCenter” on page 12 (Windows information only)
- “Installing Java” on page 27
- “Installing Ant” on page 28
- “Setting Environment Variables” on page 28
- “Documenting Your Environment” on page 29

If using the Tomcat application server in a development environment, also review:

- “Configuring the Application Server” on page 13
- “Configuring Tomcat” on page 17

If using an Oracle or SQL Server database server in a development environment, also review:

- “Configuring the Database” on page 20

After reviewing the relevant development environment information, proceed to “Installing a ClaimCenter Development Environment” on page 31.

Creating Accounts to Run ClaimCenter

It is important that the software processes that support your ClaimCenter application run with the appropriate permissions. How you set up these accounts depends on the application server environment — UNIX-based or Windows.

If your network servers are Windows systems, create a user with the **Log on as a service** right. Ensure that this user is not a member of any groups. Then, start the application server process as this user. This ensures that ClaimCenter is run with the correct rights.

Note: If you run Tomcat on Microsoft Windows, install the ClaimCenter server as a Windows service. See “Installing Tomcat as a Windows Service” on page 18 for more information.

For a UNIX-based operating system (AIX or Linux), the ClaimCenter-related processes must run in non-privileged (user) mode. A process in non-privileged mode can access only its own memory. To ensure the ClaimCenter processes run in the correct mode, create a specific user account on each server and run the corresponding applications under these accounts.

Configuring the Application Server

The ClaimCenter application server relies on a third-party servlet container for execution and connection services. This topic includes some adjustments to supported application servers.

Guidewire supports JBoss, Tomcat, WebLogic and WebSphere for production environments. Guidewire supports the use of the bundled QuickStart application server or Tomcat for development environments. Guidewire also supports JBoss, WebLogic and WebSphere application servers for development use. However, these application servers do not reload resources modified in Studio without a rebuild and redeploy. Therefore, these application servers are not ideal for development work.

See the *Guidewire Platform Support Matrix* for information about which specific application server versions Guidewire supports for ClaimCenter 6.0.8. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.

For information on configuring the QuickStart application server, see “QuickStart Application Server” on page 34.

This topic includes the following:

- “General Guidelines for the Application Server” on page 13
- “Considerations for a Clustered Application Server Environment” on page 14
- “Configuring the JVM for Environments Without Graphics” on page 15
- “JVM Heap Size Considerations” on page 15
- “Load Balancers” on page 16
- “Optional Components” on page 16
- “Configuring JBoss” on page 16
- “Configuring Tomcat” on page 17
- “Configuring WebLogic” on page 19
- “Configuring WebSphere” on page 19

General Guidelines for the Application Server

Do not include spaces in the installation path of the application server.

Run the application server on hardware supported by the application server provider. Guidewire can provide assistance with hardware requirements for your production implementation.

Only use the JDK or JRE specified in the *Guidewire Platform Support Matrix* or a higher maintenance release. Tomcat requires the JRE only.

Do not run multiple Guidewire applications or multiple instances of a single Guidewire application under a single JVM in a production environment. Guidewire does not support this configuration. Each Guidewire application in a production environment must run in a JVM reserved for that application.

ClaimCenter synchronizes with the database clock. The application server and database server must be within the same time zone. The maximum difference allowed between the application server and database server is 29 minutes.

Considerations for a Clustered Application Server Environment

This topic includes information for clustered environments. Also review “Managing Clustered Servers” on page 81 in the *System Administration Guide*.

Disabling IPv6 in Clustered Environments

Some JDKs do not function correctly with IPv6. Disable IPv6 on any application server hosting Guidewire applications. To disable IPv6, set the following java option for your application server JVM:

```
java.net.preferIPv4Stack=true
```

With Tomcat, add this option to the `CATALINA_OPTS` environment variable.

With WebLogic, either add this option to the `JAVA_OPTIONS` environment variable or directly modify the `setDomainEnv.sh` file for the domain hosting ClaimCenter. If you modify `JAVA_OPTIONS`, the option applies to all WebLogic instances on that server. If you modify `setDomainEnv.sh`, the option only applies to that domain. If you modify `setDomainEnv.sh`, add the option to the line:

```
JAVA_OPTIONS="${JAVA_OPTIONS} ${JAVA_PROPERTIES} -Dwlw.iterativeDev=${iterativeDevFlag}
-Dwlw.testConsole=${testConsoleFlag} -Dwlw.logErrorsToConsole=${logErrorsToConsoleFlag}"
```

With WebSphere, add this option using the Administrative Console. Navigate to **Servers** → **Application servers** → ***server***. In the Server Infrastructure section, click **Java and process management** → **Process definition** → **Java virtual machine** → **Custom Properties**. Then add the `java.net.preferIPv4Stack=true` option.

For more information on setting JVM options, see the documentation provided with your application server.

Configuring RedHat for Cluster Communication

In a clustered application server environment, ClaimCenter uses JGroups for some multicast communication between servers within the cluster. In a clustered RedHat environment, the default network setup does not enable proper multicast communication within the cluster.

The fix is relatively easy. Remove any line in `etc/hosts` that associates a computer by name to a `127.x.x.x` address. Then, add new lines to `etc/hosts`, using the following format:

```
127.0.0.1 localhost
COMPUTER_IP_ADDRESS COMPUTER_NAME.DOMAIN_NAME COMPUTER_NAME
```

This issue might occur with other Linux platforms. Consult the documentation for your operating system for assistance if you have an issue.

Configuring Clustering Parameters

The `config.xml` file includes several parameters to configure a clustered environment, including:

- `ClusteringEnabled`
- `ClusterMulticastAddress`
- `ClusterMulticastPort`

- `ClusterMulticastTTL`
- `ClusterProtocolStack`
- `ClusterProtocolStackOption1`
- `ClusterProtocolStackOption2`
- `ConfigVerificationEnabled`
- `PDFMergeHandlerLicenseKey`

See “Clustering Parameters” on page 46 in the *Configuration Guide* for details.

Configuring the JVM for Environments Without Graphics

For Unix and Linux environments that do not have graphics support such as an X11 graphics environment, set the Java Virtual Machine (JVM) to run in headless mode. Specify this mode by setting the `java.awt.headless` JVM parameter to `true` on your application server. This prevents the JVM from attempting to access the native graphics environment that does not exist.

JVM Heap Size Considerations

The heap size determines how much memory the Java Virtual Machine (JVM) allocates to store an executing Java program. Guidewire applications are memory intensive. Optimize performance by using large heaps.

The following sections provide instructions for increasing the JVM heap size:

- “Changing Heap Size in Tomcat on Windows” on page 18
- “Increasing Heap Size for WebLogic” on page 19
- “Increasing Heap Size for WebSphere” on page 19

Refer to the documentation provided with your application server for more information.

32-Bit Versus 64-Bit Applications

32 and 64-bit refers to the size of the pointer used to reference an address.

Production environments must use a 64-bit operating system and 64-bit JVM. 64-bit JVMs inherently use more memory to host the same number of objects. 32-bit JVMs have an inherent memory scalability limit. This limit differs significantly across platforms. This scalability limit makes those platforms unsuitable production platforms. 64-bit JVMs are a more scalable option.

Typically, a 64-bit JVM has approximately an 80% heap size overhead. For example, a 1024 MB heap for a 32-bit JVM would host the same amount of objects as a 1843 MB heap for a 64-bit JVM. Generally, non-production systems work correctly with an heap size of respectively 1024MB for a 32-bit JVM and 2048 MB for a 64-bit JVM.

Having said that, the following tables listing limits on heap size serve only as a starting point. They are useful if you are installing ClaimCenter and want to start development work. However, production systems require careful sizing. Consult Guidewire Services for assistance.

For more information on this and tuning your production application to optimal performance, contact *Guidewire Support*.

Operating System Limits on Heap Size

Operating system heap size limitations are grouped by application server.

For JBoss, Tomcat, and WebLogic

Operating system	32-bit heap size scales to:	64-bit heap size scales to:
Linux	2.7GB	Very large

Operating system	32-bit heap size scales to:	64-bit heap size scales to:
Windows 2003	1.5 GB	Very large

For WebSphere

Operating system	32-bit heap size scales to:	64-bit heap size scales to:
AIX	2 GB	Very large
Linux	2.56 GB	Very large
Windows 2003	1.5 GB	Very large

> **Note:** Although IBM recommends that the initial Java heap size for WebSphere not be set equal to the maximum Java heap size, Guidewire recommends otherwise. The IBM recommendations are not optimal for ClaimCenter. With a fixed heap, you avoid performance penalties from resizing the heap on the rising edge as the system load rises, or on the falling edge as load drops off. WebSphere provides several garbage collection policies. Guidewire recommends using the generational concurrent (gencon) garbage collection policy with equal minimum and maximum heap size.

To avoid performance degradation caused by forcing the JVM to adjust between two heap size values at runtime, set the `initial` and `maximum` values the same. The following table provides recommended heap settings for testing (determined with single user scenarios in mind). For production, consult Guidewire Services.

Single User Testing Heap Size

JVM parameter	Variable name	32-bit value	64-bit value
`inital heap size`	`Xms`	1 GB	2 GB
`maximum heap size`	`Xmx`	1 GB	2 GB
`maximum permanent generation size`	`MaxPermSize`	128 MB	256 MB

> **Note:** There is some variance across JVM technology with regard to memory allocation. Guidewire supports WebSphere on the IBM JVM only. The IBM JVM manages the permanent space without the use of a permanent size setting.

Use the heap size settings as the starting point for tuning optimal JVM settings for your configuration. Since each deployment needs to be tuned for its dataset, the heap sizes depend on your usage and configuration.

Load Balancers

In general, Guidewire supports load balancing of user interface server requests to the extent that load balancers support session affinity. It is more difficult to load balance SOAP calls because the SOAP standard does not provide a standard way to track session state across requests. However, some load balancers support IP affinity, which allows for very coarse load balancing of SOAP requests on a per-system basis. ClaimCenter supports both sessionless and sessioned SOAP calls, the latter of which require IP affinity.

Optional Components

If you plan to have ClaimCenter send email through business logic, have an SMTP-compatible email server, such as Microsoft Exchange or UNIX Sendmail available. It is also helpful to have access to an SNMP-compatible system monitoring tool, such as IBM Tivoli or HP OpenView.

Some internal monitoring tools are built into ClaimCenter. Beyond that, you can use any SNMP-compatible system monitoring tool, such as IBM Tivoli or HP OpenView if you are running on an x86/Tomcat platform.

Configuring JBoss

This section provides notes on installing JBoss to run ClaimCenter.

Removing JBoss JARs

Remove `lib\endorsed\stax-api.jar` from your JBoss installation. This JAR conflicts with ClaimCenter in the JBoss class loader.

Remove the `lib\endorsed\xercesImpl.jar` file from your JBoss installation to avoid SAX parser exceptions.

Configuring Tomcat

This section provides notes on installing Tomcat to run ClaimCenter. Install Tomcat as a system administrator.

Tomcat Examples

Guidewire recommends that you remove the `TOMCAT_HOME\webapps\examples` directory from any production Tomcat implementation. Some versions of the example scripts shipped with Tomcat have had security vulnerabilities reported.

Tomcat Native Library

Guidewire recommends that you do not implement the Tomcat Native Library. The major pitfall of using the library is that it mixes Java code and C/C++ code in the same process. This requires the use of Java Native Interface (JNI), which is not optimal for performance. Guidewire has observed performance degradation in tests with the Tomcat Native Library implemented.

The primary intent of this library is to execute some capabilities in native code versus Java. One such capability is encryption, which is calculus intensive and not optimally suited for Java. If you want to use encryption, consider offloading this task to a dedicated component such as a hardware appliance, Apache Web Server or Microsoft Internet Information Server. These components are designed for such capabilities and are therefore more secure. Additionally, having a dedicated component enables you to build a more secure network organization with a DMZ.

The Tomcat server reports a message similar to the following upon startup if the Tomcat Native Library is not implemented:

```
INFO: The Apache Tomcat Native library which allows optimal performance in production
environments was not found on the java.library.path:
```

Ignore this message.

Increasing the Maximum Concurrent Threads

By default, a Tomcat connector allows a maximum of 40 concurrent threads. Guidewire recommends that you set the maximum number of concurrent threads to 200 instead.

To increase the maximum concurrent threads

1. Open the `conf/server.xml` file in the Tomcat installation directory.
2. Find the definition for the http connector. It looks similar to the following:

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

3. Add the `maxThreads="200"` setting to this definition as follows:

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443"
           maxThreads="200" />
```

4. Save `server.xml`.
5. Restart Tomcat to make these changes effective.

Disabling Session Persistence

By default, Tomcat is configured with persistent sessions, which means Tomcat will write to disk all HTTP sessions which are in memory at the time the server is shut down.

When you restart the Tomcat server, it tries to restore the sessions. However, the session contents are only meaningful to the old instance of ClaimCenter, so ClaimCenter throws an exception such as the following:

```
SEVERE: IOException while loading persisted sessions: java.io.InvalidObjectException: Error
  deserializing Key of "com.guidewire.commons.entity.Key":null
  java.io.InvalidObjectException: Error deserializing Key of "com.guidewire.commons.entity.Key":null
  at com.guidewire.commons.entity.Key.readResolve(Key.java:141)
  ...
```

You can configure Tomcat to disable session persistence.

To disable session persistence

1. Open the Tomcat `conf/context.xml` file in a text editor.
2. Uncomment the `<Manager>` element:

```
<Manager pathname="" />
```

3. Save `context.xml`.

Installing Tomcat as a Windows Service

If you plan on installing only one application, and using Tomcat, you have the option of using the Microsoft Window Service Installer.

The Windows Service Installer automatically configures Tomcat to run as a Windows Service. If you configure ClaimCenter to run in Tomcat, Tomcat automatically runs as a Windows service. You can then set the server to start automatically as Windows starts and use the standard Windows service management tools to manage the ClaimCenter server.

> **Note:** Before the ClaimCenter application starts, the database server must already be up and running. Keep this order issue in mind as you develop startup procedures and scripts.

To access additional configuration parameters while running Tomcat as a Windows service, use Tomcat Service Manager. This is a GUI-based application designed to make it easy to configure, install, and manage one or more instances of Jakarta Tomcat running as Windows services. This program is available at `http://web.bvu.edu/staff/david`.

Changing Heap Size in Tomcat on Windows

Set heap size for Tomcat by setting a Windows environment variable.

To increase the Tomcat heap size on Windows

1. From the Windows desktop, right click **My Computer**.
2. Select **Properties** and click the **Advanced** tab.
3. Click the **Environment Variables** button.
4. Under System Variables, click **New**.
5. Set the **Variable Name** to `CATALINA_OPTS`.
6. Set the **Variable Value** to `-Xms1024m -Xmx1024m`.

7. Click **OK** until the properties settings closes.

   Note: For values in step 6, consult “JVM Heap Size Considerations” on page 15.

Configuring WebLogic

Review this topic if you are using WebLogic as the application server.

Increasing Heap Size for WebLogic

Use the `USER_MEM_ARGS` environment variable to specify arguments to WebLogic during application server startup.

To increase the WebLogic heap size

1. Create an environment variable on your system named `USER_MEM_ARGS.`
2. For the value of `USER_MEM_ARGS`, enter:

   ```
   -Xms256m -Xmx1024m -Dgw.server.mode=dev
   ```

   Note: The server mode entry is needed for reloading PCF pages. If you want to run the server in production mode, remove the `-Dgw.server.mode` argument. By default, ClaimCenter starts in production mode on all application servers except the bundled QuickStart server.

Configuring WebSphere

Review this topic if you are using WebSphere as the application server.

Increasing Heap Size for WebSphere

Increase minimum heap size to 256 and maximum heap size to 1024.

To increase the WebSphere Heap Size

1. Open the WebSphere Administrative Console.
2. From the left menu, select **Servers** → **Server Types** → **WebSphere application servers**, and select your server from the list on the right.
3. Under **Server Infrastructure** select **Java and Process Management** → **Process Definition.**
4. Under **Additional Properties**, select **Java Virtual Machine.**
5. Enter the following heap sizes, then click **OK**.
   - For **Initial heap size** enter: `256`
   - For **Maximum heap size** enter `1024`
6. Click **OK**.
7. A message displays stating that changes have been made. Click **Save**.

Adjusting Ping Parameters for WebSphere with Oracle

If WebSphere times out while communicating with Oracle, set the following values:

Server Parameter	Value
`ping interval`	2000
`ping timeout`	6000

WebSphere Database Cluster Management

Guidewire supports using WebSphere database cluster management, provided that you use the JDBC JAR file bundled with ClaimCenter.

WebSphere Network Deployment

Guidewire certifies the base version of WebSphere, but supports WebSphere Network Deployment (ND). Guidewire products do not require any ND functionality to run. Some IBM terminology conflicts with Guidewire terminology. These differences are clarified as follows:

ClaimCenter provides *cache coherency (clustering)* since it maintains an internal cache of objects for performance reasons. After ClaimCenter changes an object in the cache for a single node in a cluster, ClaimCenter sends a message to invalidate the object in other node caches. The Guidewire clustering for cache coherency implementation is completely independent of the application server.

WebSphere ND provides three key features that you might like to use in conjunction with ClaimCenter.

- **Deployment**. WebSphere ND includes tools for automatically deploying configurations to servers in a cluster. Guidewire supports this if used in conjunction with our Environment Specific Configuration settings.
- **Load balancing**. WebSphere ND provides load balancing capabilities. See “Load Balancers” on page 16.
- **Clustering**. This is different from the ClaimCenter cache coherency capability. WebSphere ND clustering is mostly about session replication, which is the capability to constantly maintain a user’s session state across multiple computers in a cluster. This is useful in the event of failover, in which WebSphere transfers a user from one server to another. This functionality was intended to support lightweight session objects such as those found in online shopping carts. Because Guidewire designed ClaimCenter for enterprise users, each session must preserve a large volume of data. As a result, session replication is prohibitively performance intensive and Guidewire does not support it.

Configuring the Database

Guidewire recommends that you implement the guidelines contained in this topic while you install and configure the database server. ClaimCenter 6.0.8 supports Oracle and SQL Server for production environments. See the *Guidewire Platform Support Matrix* for information about which specific database server versions Guidewire supports for ClaimCenter 6.0.8. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.

Run the database server on hardware supported by the database server provider. Guidewire can provide assistance with hardware requirements for your production implementation.

For production systems, Guidewire recommends using a database server dedicated to ClaimCenter. Production environments must use a 64-bit operating system and 64-bit database engine on the database server.

ClaimCenter synchronizes with the database clock. The application server and database server must be within the same time zone. The maximum difference allowed between the application server and database server is 29 minutes.

ClaimCenter depends heavily on back-end database performance, which in turn depends on storage performance. Optimize the performance of your database server.

This topic includes:

- “Database Permissions” on page 21
- “Configuring Linguistic Search Collation” on page 21
- “Configuring Oracle for ClaimCenter” on page 21
- “Configuring SQL Server for ClaimCenter” on page 23

Database Permissions

Create a user called `ccUser` in your database. ClaimCenter connects to this user. The `ccUser` must have the correct permissions. The following table lists the permissions required for each database:

Database	Permissions Required
Oracle	The `ccUser` must have the following permissions on the ClaimCenter database: • create session • create procedure • create trigger • create table • create view • create sequence • query rewrite • alter session • select any dictionary If your users want to see statspack data on the ClaimCenter **Info Pages** interface, grant the `ccUser` access to the ClaimCenter performance statistics (`perfstat`) tables.
SQL Server	The `ccUser` must have the `public` and `db_owner` roles on the ClaimCenter database. ClaimCenter supports several different data management pages for performance analysis of the application. To use these pages, the `ccUser` must be granted `view server state`. The server login account must also have `view database state` permission on each ClaimCenter data management view. The data management views all start with `sys.dm_` prefix.

Configuring Linguistic Search Collation

You can configure how ClaimCenter searches and sorts search results. For example, you can configure whether searching is accent-sensitive or accent-insensitive, case-sensitive or case-insensitive. For more information, see “Localized Search and Sort” on page 525 in the *Configuration Guide*.

Configuring Oracle for ClaimCenter

These directions assume you have already installed Oracle. See the *Guidewire Platform Support Matrix* for information about which specific Oracle versions Guidewire supports for ClaimCenter 6.0.8. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.

Prerequisites to Installing on Oracle

Ensure that your Oracle database is running in an environment that optimizes storage performance while maintaining storage maintainability. To achieve this goal, Oracle recommends you use the SAME (Stripe and Mirror Everywhere) strategy. To learn more about this strategy, consult the following `http://www.oracle.com/technology/deploy/performance/pdf/opt_storage_conf.pdf`.

Configure Oracle to use asynchronous IOs. Asynchronous IO significantly simplifies a database’s IO management while increasing its performance. For performance reasons, tune your operating system to Oracle database requirements. Oracle provides guidance on tuning your database based on:

- operating system
- available memory
- database release

Consult Oracle documentation and support web site for information on how to tune your database.

The Oracle database supports server-side caching that can help increase ClaimCenter performance. The size of the Oracle database cache is critical to supporting this feature. For internal tests, Guidewire uses a database cache size of 3.6 GB or more. Consult the Oracle documentation for information on selecting a cache size appropriate to your server computer's architecture.

After your database is in production, you can not easily modify the storage architecture. Guidewire recommends that you test and tune your database's storage performance prior to installing ClaimCenter. There are many tools available for this task including an open source tool, IOzone (`www.iozone.org`).

To prepare an Oracle database for ClaimCenter

1. Create a new database instance for ClaimCenter.

   Guidewire recommends that you **not** share the ClaimCenter database with other data or applications.

   IMPORTANT Guidewire currently supports single-byte character sets that are a strict superset of ASCII, and `AL32UTF8` or `UTF8` for Unicode. Only use a supported character set with ClaimCenter. Refer to your Oracle documentation for a complete list of supported character sets. `WE8ISO8859P1` is a single-byte character set that supports both Western European languages and American English. `AL32UTF8` and `UTF8` are Oracle character sets supported for the storage of Unicode data, such as Asian characters. If using the `AL32UTF8` or `UTF8` character set, the Oracle instance must be configured with `nls_length_semantics` set to `char`. Otherwise, the database does not start.

2. Create one or more tablespaces to support the ClaimCenter logical tablespaces. Guidewire recommends that you create a separate tablespace for each logical tablespace:

Logical Name	Usage
ADMIN	Stores system parameters.
OP	Stores the main ClaimCenter data tables.
TYPELIST	Stores system code tables.
INDEX	Stores system indexes.
STAGING	Stores inbound staging data tables.

   You can name your tablespaces anything you like: either the same as the logical tablespace names or entirely different. As you configure ClaimCenter, you map the tablespaces you created to the ClaimCenter internal logical tablespaces.

3. Create a single database user, `ccUser,` within the ClaimCenter database.

4. Grant `ccUser` the following permissions:
   - alter session
   - create procedure
   - create sequence
   - create session
   - create table
   - create trigger
   - create view
   - query rewrite
   - select any dictionary

   If your users want to view statspack data on the ClaimCenter Info Pages interface, you also need to grant the `ccUser` access to Statspack's (perfstat user) tables.

5. Grant `quota` on all the tablespaces listed in step 2 to the `ccUser`.

6. Set default tablespace for `ccUser` to the one being mapped to the `OP` logical tablespace.

7. If you run the database server and the application server on the same computer, be aware that Oracle adds directories to the `PATH` environment variable. To prevent potential conflicts with ClaimCenter files, Guidewire recommends that you edit your `PATH` variable and move the Oracle directories to the end, after the ClaimCenter directories. Guidewire recommends that you do not run database and application servers on the same computer in a production environment.
8. Test a connection to the database from a database client. Verify that all the tablespaces are visible.

Using Oracle Resource Consumer Groups for Slow Claim Queries

Note: The information in this section is provided for implementations that experience slow Claim queries. If you do not experience slow Claim queries, you do not need to define a resource consumer group for Claim queries.

Oracle provides resource plans and consumer groups to handle resources in the database. One useful feature is to cancel a query based on the execution time. You can configure ClaimCenter to switch to a resource consumer group that you define to perform Claim searches. You can set this resource consumer group to have a time limit and cancel SQL operations for Claim searches that exceed the time limit.

ClaimCenter saves the initial resource consumer group detected when the application server is started and reverts to that group following the Claim query.

The requirements for this functionality to work are:

- The ClaimCenter parameter `UseDbResourceMgrCancelSql` is set in `config.xml` to a resource group defined in Oracle.
- The Oracle resource manager plan is set at the system level.
- The `ccUser` has privileges to switch between the two resource groups.

ClaimCenter checks these conditions when the server starts.

This feature is currently applicable only to Oracle.

Configuring SQL Server for ClaimCenter

This topic includes information on how to configure SQL Server for ClaimCenter. See the *Guidewire Platform Support Matrix* for information about which specific SQL Server versions Guidewire supports for ClaimCenter 6.0.8. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.

Guidewire does not support Windows Integrated Security as an option while connecting to SQL Server.

Prerequisites to Installing on SQL Server

IMPORTANT Guidewire requires the SQL Server database to be case-insensitive. During startup, ClaimCenter checks that the SQL Server database is case-insensitive.

Decide whether to store all character data in single-byte format in `varchar` columns, or Unicode multi-byte format stored in `nvarchar` columns. If using Unicode, such as for Asian languages, then the `database` element in `config.xml` must set the `unicodecolumns` attribute to `true`. Then, when the database tables are created the first time the application server is started, all character columns are created with the `nvarchar` datatype.

The collation setting specifies sorting and comparison rules, and also the code page to use for single-byte data. Microsoft still supports SQL Server collations (that start with `SQL_`) in addition to Windows collations, but recommends using a Windows collation. Guidewire also recommends that you use a Windows collation. Choose the collation setting carefully. Refer to Collation and International Terminology at `http://msdn.microsoft.com/en-us/library/ms143726.aspx` for a full discussion. The version of Windows being used for the database and application server is a factor. Some newer collations, such as `Japanese_Bushu_Kakusu_100`, are only available on Windows 2007 or later and not on Windows 2003.

Guidewire requires that the collation specifies `CI`, or case-insensitive. The case-sensitivity setting affects table and column names, and Guidewire requires these names to be case-insensitive.

Creating a SQL Server database with files of sufficient size and parameters is important to future performance and maintenance. A basic discussion of this can be found online in a Microsoft SQL Server topic "Designing Databases" at `http://msdn.microsoft.com/en-us/library/ms187099.aspx?ppud=4`.

For production systems Guidewire strongly recommends that you pre-allocate disk space rather than using the SQL Server `autogrowth` feature. As a general guideline, estimate how big your database might grow in one year and add 20%. Then, allocate enough total file space for this size. Monitor the size of the database and add space during scheduled periods of lower activity. Set the maximum file size to be less than the size of the disk, so that the disk does not fill up.

For your production database, work with your SAN (Storage Area Network) engineers early in implementation to deliver production-realistic performance.

Guidewire recommends that you **not** share the SQL Server instance on which you are running ClaimCenter with other data or applications.

To install ClaimCenter on SQL Server

1. Configure SQL Server. See "Configuring SQL Server in Management Studio" on page 24.
2. Create a database for ClaimCenter. See "Creating a ClaimCenter Database in SQL Server" on page 25.
3. Modify the `config.xml` file so that the application correctly points to the database. See "Deploying ClaimCenter to the Application Server" on page 56.
4. Restart the application server and test by opening ClaimCenter in a browser window.

Configuring SQL Server in Management Studio

To configure SQL Server to support ClaimCenter

1. Open SQL Server Management Studio.
2. In the **Object Explorer**, right-click the server node you plan to use for ClaimCenter and choose **Properties**. Typically, the node is the same as computer name.

   The **Server Properties** dialog opens.
3. Select the **Security** page and check **SQL Server and Windows Authentication Mode**.

   WARNING ClaimCenter does not run if authentication is set to **Windows Authentication Mode** only.
4. Select the **Memory** page.
5. Adjust the **Maximum Server Memory** to use at least 200 MB.
6. With a dedicated host computer running SQL Server, Microsoft recommends that you use the default settings and have SQL Server manage memory. Consult the Microsoft documentation (`http://msdn2.microsoft.com/en-us/library/ms178067(d=ide).aspx`) for an in-depth discussion of memory options. Setting the maximum server memory to a particular value can cause performance problems.
7. Click **OK** to close the dialog.
8. Right-click the **SQL Server Agent** node and select **Properties**.
9. Select the **General** page.

10. Check **Autostart SQL Server if it stops unexpectedly.**

11. Click **OK** to close the dialog.

Creating a ClaimCenter Database in SQL Server

This topic includes procedures to create and configure a SQL Server database for ClaimCenter.

IMPORTANT If you plan to create additional database instances to support multiple ClaimCenter environments or other Guidewire products, consider applying the changes in the following procedure to the model database. Use the model database as a template for the additional database instances. Before you edit the model database, create a backup.

To create and configure a SQL Server instance for ClaimCenter

1. If you have not already done so, open SQL Server Management Studio. If you are creating a new database, proceed to step 2.

   If you are modifying the model database, expand **Databases** → **System Databases**, right-click **model** and select **Properties**, and skip to step 4.

2. Right-click the **Databases** node and select **New Database**. Or, your company's database administrator can write a `CREATE DATABASE` SQL statement to create the database.

   Guidewire recommends that you not share the ClaimCenter database with other applications.

3. Enter a database name in the **New Database** dialog and click **OK**.

4. Optionally, create one or more filegroups to support the ClaimCenter logical tablespaces from the **Filegroups** page. If you choose to use filegroups, create a separate filegroup for each of the following logical tablespaces:

Logical Name	Usage
ADMIN	Stores system parameters.
OP	Stores the main ClaimCenter data tables.
TYPELIST	Stores system code tables.
INDEX	Stores system indexes.
STAGING	Stores inbound staging data tables.

   With one exception, you can name the filegroups anything you like: either the same as the logical tablespace names or entirely different. `INDEX` is a reserved name on SQL Server, so you can not map the logical tablespace `INDEX` to a physical filegroup of the same name.

   As you configure the ClaimCenter database connection, you map the filegroups you created to the ClaimCenter internal logical tablespaces.

5. Select the **Options** page.

6. Choose your database collation if not using the SQL Server server default. The only requirement is that it is a CI (case-insensitive) collation.

7. Validate that `Auto Create Statistics` and `Auto Update Statistics` are both set to `True`. During startup, ClaimCenter checks that these properties are set to `True` and validates that the SQL Server database is case-insensitive.

8. Validate that `Auto Shrink` is set to `False`. If set to `True`, poor performance can result.

9. Click **OK**.

10. Right-click **Security** and select **New** → **Login**.

11. On the **Login - New** dialog, select **SQL Server Authentication** if not already selected.
12. Specify a password and password policy options.
13. Click **OK**.
14. In **Object Explorer**, expand the database and open **Security** → **Users**.
15. Right-click **Users** and select **New User.**
16. Enter `ccUser` for the **User name**.
17. Enter the **Login name** that you created earlier.
18. Grant this user ownership of the ClaimCenter database by selecting `db_owner` in both **Schemes owned by this user** and **Database role membership** panels.
19. Click **OK**.
20. ClaimCenter supports several different data management pages for performance analysis of the application. To use these pages, the `ccUser` must be granted `view server state` and `view database state` on each ClaimCenter data management view. The data management views all start with `sys.dm_ prefix`.
    a. Right-click the database and select **Properties**.
    b. Select the **Permissions** page.
    c. Select `ccUser`.
    d. Select the checkbox to grant `view database state` permission.
    e. Click **OK**.
    f. Right-click the server and select **Properties**.
    g. Select the **Permissions** page.
    h. Select the login associated with ccUser.
    i. Select the checkbox to grant `view server state` permission.
    j. Click **OK**.
21. Guidewire recommends that you do not use the SQL Server `autogrowth` feature in a production system. Instead, monitor the size of your database and increase the size of the database files as needed during periods of lower activity. SQL Server enables the `autogrowth` feature by default.

    To disable autogrowth

    a. Right-click the database and select **Properties**.
    b. Select the **Files** page.
    c. For each database file, click the **...** button in the **Autogrowth** column.
    d. Click the checkbox for **Enable Autogrowth** to deselect it.
    e. Click **OK**.
    f. Repeat step b through step e for each database file.
    g. Click **OK** on the **Database Properties** screen.
22. ClaimCenter requires that the `READ_COMMITTED_SNAPSHOT` option is on.

    To set this parameter

    a. Click **New Query.**

b. In the query pane, enter:

```
ALTER DATABASE dbname
SET READ_COMMITTED_SNAPSHOT ON
WITH ROLLBACK IMMEDIATE
GO
```

c. Click **Execute**. SQL Server Management Studio informs you that the command completed successfully.

During startup, ClaimCenter checks that the `READ_COMMITTED_SNAPSHOT` option is on.

IMPORTANT The use of `READ_COMMITTED_SNAPSHOT` greatly increases resource requirements on the `tempdb` database. Set `tempdb` to grow in 10% increments, and provide sufficient disk space for `tempdb` to grow substantially. Performance can be improved if you dedicate separate I/O resources to `tempdb`.

23. Close SQL Server Management Studio. You do not need to save the `READ_COMMITTED_SNAPSHOT` query.

Before Continuing...

Check that your SQL server environment is correct. Test that you can connect to the database using the `ccUser` credentials. Ensure that SQL Server starts automatically as the server starts. To test if your database is starting automatically, reboot your server and attempt to access the database from a client.

Development Workstation Information

Each developer workstation is a contained environment that includes your company's Guidewire applications and the components needed to configure it. Guidewire recommends that development workstation and environment meet requirements beyond end-user workstations.

Guidewire recommends high-performance I/O systems for development, such as RAID-0 and SCSI or SSD disks. Studio is a high I/O application. Installation of software that slows the I/O system, such as encryption or continuous virus scans can handicap development productivity.

See the *Guidewire Platform Support Matrix* for development workstation system requirements for ClaimCenter 6.0.8. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.

Client Information

ClaimCenter is a web application accessed through a web browser. See the *Guidewire Platform Support Matrix* for client system requirements for ClaimCenter 6.0.8. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.

Installing Java

The ClaimCenter application server and Guidewire Studio require a JVM (Java Virtual Machine). The version of the JVM depends on the servlet container and operating system on which the application server runs. See the *Guidewire Platform Support Matrix* for specific version requirements. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.

IMPORTANT Production environments must use a 64-bit operating system and 64-bit JVM.

To use a 64-bit Oracle JDK for development, add the startup parameter `-XX:+UseCompressedOops` to the JVM.

By default, Oracle JVMs provide both a client and a server mode. Guidewire supports only the server mode as it yields much higher performance. How you set this mode depends on your application server.

- If using Oracle JVM with Tomcat, then add the `-server` flag to `CATALINA_OPTS`.
- If using Oracle JVM with WebLogic, then add the `-server` flag as an argument while launching the WebLogic start script.
- If using the IBM JVM with WebSphere, the server mode is enabled by default. You probably do not need to change any settings.

Refer to `http://www.oracle.com/technetwork/java/javase/downloads/index.html` for information on downloading the JDK.

Installing Ant

Ant is a common tool used for platform-independent scripting with Java systems. The ClaimCenter configuration environment and administration tools use Ant for various system administration scripts. Install Ant on each computer that runs these scripts.

See the *Guidewire Platform Support Matrix* for specific Ant version requirements. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.

Obtain Ant from the following URL:

```
http://ant.apache.org/bindownload.cgi
```

If you have issues running Guidewire `gwcc` commands, remove any `-XX` options from the `ANT_OPTIONS` environment variable to determine if there is a conflict with non-standard options.

Setting Environment Variables

After you install Java and Ant, set environment variables so that ClaimCenter can locate them. Make these environment variables available in the user environment in which you plan to run ClaimCenter. The following table lists the variables to set for the different systems:

System	Variable	Example Values and Notes
Application server (all)	`JAVA_HOME`	`C:\Program Files\Java\jdk1.6.0_24` The Java installer sets `JAVA_HOME` automatically, but Guidewire recommends that you verify that it is set correctly.

Application server (Tomcat)	`CATALINA_OPTS`	Specifies the minimum and maximum memory used by Tomcat. For example, the following value for `CATALINA_OPTS` would set direct JVM memory allocations to 1024 MB (initially) and 1024 MB (maximum), and would allocate 128 MB of background processing memory:
		`-server -Xms1024M -Xmx1024M -XX:PermSize=128m -XX:MaxPermSize=128M`
		Make your maximum JVM memory allocation (the `–Xmx` setting) the maximum likely available memory on the server. Guidewire tests have shown that performance of garbage collections are best if the `-Xms` and `-Xmx` are set to the same value. See "Operating System Limits on Heap Size" on page 15 for a detailed discussion.
		For more information on configuration options, run the following command to view the built-in help for Java command line options:
		`java –X`

Development environment, administration tools	`JAVA_HOME`
	`ANT_HOME`
	`Add Ant to Path`
		`C:\Program Files\Java\jdk1.6.0_24`
		`C:\ant`
		`C:\ant\bin`

Before Continuing

Check that you established your environment correctly. Open a new command window and display your environment variables to check them.

Documenting Your Environment

Now that you have established your environment, take some time to document it in preparation for installing ClaimCenter. Enter your configuration values in the following tables:

Database Configuration	Value
Database Name	
Server Name	
Database server port	
ClaimCenter database user name	
Cache size	
Block size	

Application Server Environment Variables	Value
The user name application runs under	
`ANT_HOME`	
`JAVA_HOME`	
`CATALINA_HOME`	
`CATALINA_OPTS`	

chapter 3

Installing a ClaimCenter Development Environment

ClaimCenter supports a variety of development environment options depending on your business needs. You can:

- Perform development work using the default bundled QuickStart application server. To deploy a QuickStart development environment, see "Installing the QuickStart Development Environment" on page 32.
- Configure Tomcat to automatically load configuration changes that you make in Guidewire Studio. For instructions, see "Installing a Tomcat Development Environment" on page 35.
- Work on the local configuration files, repackage the configured application into a WAR or EAR file, and deploy it to a local or remote application server. Because you must repackage and redeploy a WAR or EAR file after making configuration changes, Guidewire does not recommend this approach for a development environment. See:
  - "Installing a JBoss Production Environment" on page 56
  - "Installing a WebSphere Production Environment" on page 59
  - "Installing a WebLogic Production Environment" on page 58

Use the QuickStart method if you want to quickly install a development or demonstration ClaimCenter environment using the fewest steps.

IMPORTANT This topic only provides information for installing a ClaimCenter development environment. To install a production environment, first review "Preparing a ClaimCenter Environment", on page 11 and then proceed to "Installing a ClaimCenter Production Environment", on page 41.

This topic includes:

- "Using Multiple ClaimCenter Development Instances" on page 32
- "Installing the QuickStart Development Environment" on page 32
- "Installing a Tomcat Development Environment" on page 35
- "Using the QuickStart Database" on page 37

- “Using SQL Server or Oracle in a Development Environment” on page 38
- “Enabling Archiving for Development Testing” on page 38
- “Installing Sample Data” on page 39

Using Multiple ClaimCenter Development Instances

Occasionally, a developer might want to connect from one machine to multiple ClaimCenter instances running on the same physical or virtual server. In this case, the port number differs for each instance, but the IP address and domain name are the same between the two application instances.

Most containers hold the session ID in a cookie. The container gives the cookie a default name and associates the cookie with a host name or IP address and a path. If you run multiple application servers for the same application, each one generates session cookies with the same host name and path. This session cookie does not include the port number. Therefore, if you log into one application instance, the browser ends the session with any other application servers having the same host and path, even if port numbers differ.

> **Note:** This is not an issue if you run two different Guidewire applications on a single machine (for example, ClaimCenter and BillingCenter). The two different applications run under different `webapp` paths.

To work around this issue, open the application instance sessions using different paths. For example, use the fully qualified domain machine name for one application server and `localhost` for the second application server. The browser does not associate the same cookie with an IP address and with a machine name.

Installing the QuickStart Development Environment

This topic describes using the ClaimCenter QuickStart development environment. The QuickStart method uses a bundled and lightweight application server and database that are suitable for development and demonstration purposes. Guidewire does not support the QuickStart method for a production environment.

This topic includes:

- “Advantages to Using the QuickStart Software” on page 32
- “QuickStart Development Environment Prerequisites” on page 33
- “Installing with QuickStart” on page 33
- “QuickStart Commands” on page 34
- “QuickStart Application Server” on page 34
- “Troubleshooting QuickStart” on page 35

Advantages to Using the QuickStart Software

The bundled lightweight application server and database provided with ClaimCenter make it possible to accomplish more work with less effort and in less time. The following are some specific benefits to using the QuickStart configuration with Guidewire Studio as the IDE (Integrated Development Environment):

- Install and run ClaimCenter rapidly without any configuration.
- Configure ClaimCenter without needing to repackage WAR or EAR files.
- Import sample data and create new data through the user interface.
- View and experiment with the default functionality of ClaimCenter.
- Make changes to ClaimCenter using Guidewire Studio.

The QuickStart application server (Jetty) is a fully certified servlet container that starts faster than production application servers. It also provides an instantaneous view of configuration changes. The QuickStart server uses the ClaimCenter configuration files from the file system, rather than requiring a packaged WAR or EAR file. Therefore, developers can configure ClaimCenter without needing to repackage the application.

> **Note:** If you do not want to use the QuickStart application server for development, you can configure Tomcat to point to the configuration resources being edited by Guidewire Studio. For instructions, consult "Installing a Tomcat Development Environment" on page 35. If you decide to use Tomcat, you must deploy a WAR file for production purposes.

QuickStart Development Environment Prerequisites

- Your environment must meet the minimum requirements for a development workstation. See the *Guidewire Platform Support Matrix* for current system and patch level requirements. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`.
- ClaimCenter requires Java and Ant. See "Installing Java" on page 27 and "Installing Ant" on page 28 for installation guidelines.

Installing with QuickStart

This topic includes a procedure to install ClaimCenter with the QuickStart application server.

ClaimCenter uses a QuickStart database by default. See "Using the QuickStart Database" on page 37. You can configure a ClaimCenter development environment to use an Oracle or SQL Server database instead. See "Using SQL Server or Oracle in a Development Environment" on page 38 for instructions.

For instructions to install a QuickStart ContactCenter development environment, see the *Guidewire Contact Management Guide*.

To install ClaimCenter with the bundled QuickStart application server

1. Create an installation directory for ClaimCenter. Do not use spaces in the installation directory path. This guide uses `ClaimCenter` as the directory name.
2. Unzip the ClaimCenter ZIP file into the ClaimCenter directory. See "ClaimCenter Configuration Files" on page 88 in the *Configuration Guide* for a list of how the files are organized.
3. If you are not using a version control system, make a read-only copy of the ClaimCenter directory. This enables you to recover quickly from accidental changes that can prevent ClaimCenter from starting.
4. Open a command window to `ClaimCenter\bin`.
5. If you are reinstalling ClaimCenter and using a QuickStart database, drop the QuickStart database by running `gwcc dev-dropdb`.
6. Run the following command:
   ```
   gwcc copy-starter-resources
   ```
   This command copies configuration resources, including `config.xml`, to a configuration module at `ClaimCenter\modules\configuration`.
7. ClaimCenter uses the QuickStart database by default. You can configure where ClaimCenter stores the QuickStart database files. See "Using the QuickStart Database" on page 37. If you want to use an Oracle or SQL Server database in your development environment, see "Using SQL Server or Oracle in a Development Environment" on page 38. Then continue with this procedure after you have created the database account and configured ClaimCenter to connect to the database.
8. Start the QuickStart server with the following command:
   ```
   gwcc dev-start
   ```
   When the server has started, you see: `*****ClaimCenter ready*****` in the command window.

9. Open a browser and navigate to `http://localhost:8080/cc` and login with the default superuser.
    - User name is `su`.
    - Password is `gw`.

After installing ClaimCenter with the QuickStart application server, see the following topics for procedures you might want to use to set up your development environment:

- "Using the QuickStart Database" on page 37
- "Using SQL Server or Oracle in a Development Environment" on page 38
- "Installing Sample Data" on page 39
- "Configuring Logging" on page 35 in the *System Administration Guide*
- *ClaimCenter Configuration Guide*
- To integrate ClaimCenter with ContactCenter, see the *Guidewire Contact Management Guide*.

QuickStart Commands

You launch many ClaimCenter commands by passing arguments to the `gwcc` command, located in `ClaimCenter\bin`. For a complete list of `gwcc` commands, see "Commands Reference", on page 69.

QuickStart Application Server

The QuickStart application server is Jetty, a Java-based HTTP Server and Servlet Container. Jetty was released as an open source project under the Apache 2.0 License and is fully-featured. Refer to `http://jetty.mortbay.com` for details. This application server is suitable for demonstration and development environments. This application server is not suitable nor supported for production environments.

ClaimCenter on the QuickStart server always runs in development mode. You cannot run ClaimCenter on the QuickStart server in production mode. For more information on server modes, see "Determining Server Mode" on page 62.

Configuring QuickStart Ports

The `ClaimCenter\modules\cc\etc\jetty.properties` file lists the ports used by the QuickStart server. You can specify the port on which the server listens, a debug port, and the port to use to stop the server.

The ClaimCenter QuickStart application server listens on port 8080 by default. The ContactCenter QuickStart application server listens on port 8280 by default.

To change one or more of these ports, copy the `ClaimCenter\modules\cc\etc\jetty.properties` file to `ClaimCenter\modules\configuration\etc`. Then edit the port values in `ClaimCenter\modules\configuration\etc\jetty.properties`.

IMPORTANT You cannot assign a port number between 8800 and 8900 to the QuickStart server.

Tuning QuickStart Application Server Memory

The `ClaimCenter\modules\pl\etc\memory.properties` file specifies memory settings for the QuickStart application server and other `gwcc` tools. You can set the starting heap size, maximum heap size and maximum permanent size for the QuickStart server. To change one or more of these settings, copy the `ClaimCenter\modules\pl\etc\memory.properties` file to `ClaimCenter\modules\configuration\etc`. Then edit the following values in `ClaimCenter\modules\configuration\etc\memory.properties`:

- starting heap size: `com.guidewire.commons.jetty.GWServerJettyServerMain.xms`
- maximum heap size: `com.guidewire.commons.jetty.GWServerJettyServerMain.xmx`
- maximum permanent size: `com.guidewire.commons.jetty.GWServerJettyServerMain.maxperm`

You can also adjust the memory settings for other `gwcc` tools. See “Tuning Command Line Tool Memory Settings” on page 69.

Troubleshooting QuickStart

If you have problems with QuickStart, consider the following:

Issue: You are unable to upgrade or to see changes after changing database tables.

Solution: First try restarting the server. If that does not work, then drop the database. If the server is running, then stop the server by opening a command window, navigating to `ClaimCenter\bin` and entering the following command:

```
gwcc dev-stop
```

Then, drop the database with the command `gwcc dev-dropdb`.

Issue: You experience port conflicts.

Solution: The QuickStart server listens on a default server port. This port might already be in use by your organization. Consult with your IT department to verify which ports to use. To change the server port, see “Configuring QuickStart Ports” on page 34.

Installing a Tomcat Development Environment

Although the QuickStart Jetty application server is suitable for most development needs, some organizations prefer to use Tomcat in their development environment. The following procedure enables you to create a ClaimCenter development environment on Tomcat in which you can make configuration changes without having to repackage the application.

To configure a Tomcat ContactCenter development environment, see the *Guidewire Contact Management Guide*.

Guidewire provides testing APIs in JAR files with names ending in `-gunit.jar` for use during configuration and development. These APIs are only available when the application is running in development mode, or from within Guidewire Studio. When WAR or EAR files are built, the testing API JAR files are excluded. If your deployed code makes calls to any testing APIs, it causes `ClassNotFoundExceptions` and other problems and prevents the application from running properly.

See also “Using Multiple ClaimCenter Development Instances” on page 32.

Tomcat Development Environment Prerequisites

- Your environment must meet the minimum requirements for a development workstation. See the *Guidewire Platform Support Matrix* for current system and patch level requirements. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at `https://guidewire.custhelp.com/app/resources/products/platform`. Also see “Development Workstation Information” on page 27.
- ClaimCenter requires Java and Ant. Guidewire strongly recommends that you also install the Dynamic Code Evolution Virtual Machine (DCE VM) on development environments. If you do not install the DCE VM, you will not be able to see dynamic changes to Gosu code. See “Installing Java” on page 27 and “Installing Ant” on page 28 for specific required versions and installation guidelines.

Building a ClaimCenter Development Environment with Tomcat

The following procedure installs a ClaimCenter development environment on Tomcat. If you want to set up a production environment on Tomcat, see “Installing a Tomcat Production Environment” on page 57

To set up a ClaimCenter development environment on Tomcat

1. If you have not already done so, create an installation directory for ClaimCenter. This guide uses `ClaimCenter` as the directory name.
2. Unzip the ClaimCenter ZIP file into the ClaimCenter directory. See "ClaimCenter Configuration Files" on page 88 in the *Configuration Guide* for a list of how the files are organized.
3. If you are not using a version control system, make a read-only copy of the ClaimCenter directory. This enables you to recover quickly from accidental changes that can prevent ClaimCenter from starting.
4. Create or modify the `CATALINA_OPTS` environment variable to include the following:
   ```
   -Xms1024m -Xmx1024m -XX:MaxPermSize=128m -Dgw.server.mode=dev
   ```
5. Open a command window to `ClaimCenter\bin`.
6. If you are reinstalling ClaimCenter and using the QuickStart database, drop the database by running `gwcc dev-dropdb`.
7. Run the following command:
   ```
   gwcc copy-starter-resources
   ```
   This command copies configuration resources, including `config.xml`, to a configuration module at `ClaimCenter\modules\configuration`.
8. ClaimCenter uses the QuickStart database by default. You can configure where ClaimCenter stores the QuickStart database files. See "Using the QuickStart Database" on page 37. If you want to use an Oracle or SQL Server database in your development environment, see "Using SQL Server or Oracle in a Development Environment" on page 38. Then continue with this procedure after you have created the database account and configured ClaimCenter to connect to the database.
9. Run the following command:
   ```
   gwcc build-war
   ```
   This command creates a `cc.war` file and places it in the `ClaimCenter\dist\war` directory.
10. Deploy the package to Tomcat by copying the `cc.war` file to the `webapps` directory in your Tomcat server.
11. Use the Tomcat `bin\startup.bat` command to start Tomcat and allow it to explode the WAR file. When Tomcat starts, it automatically recognizes this new application and unpacks the `cc.war` into a directory structure within `webapps.` For this example, Tomcat creates a `webapps\cc` directory. Each time you deploy a new copy of a `cc.war` file, delete the pre-existing `cc` directory structure created by the old `cc.war` file.
12. Delete the Tomcat `webapps\cc\modules\configuration` directory.
13. Create a symbolic link from the deployed ClaimCenter application on Tomcat to the `ClaimCenter\modules\configuration` directory of the original ClaimCenter installation location.

    Windows 7 and Vista: Use the `mklink` command to create the link, as in the example below:
    ```
    mklink /d C:\apache-tomcat-6.0.18\webapps\cc\modules\configuration
      C:\ClaimCenter\modules\configuration
    ```
    Windows XP: Use the Windows SysInternals program `Junction.exe` to create the symbolic link. This program is available at:
    ```
    http://technet.microsoft.com/en-us/sysinternals/bb896768.aspx
    ```
14. Restart the Tomcat server.

To test the ClaimCenter development environment on Tomcat

1. If not already running, start the Tomcat server.
2. Open a browser to the ClaimCenter development environment URL. For example:
   ```
   http://localhost:8080/cc
   ```
   Notice that the login page includes a **User name** field. You change this field in this procedure.

3. Launch Guidewire Studio by running the following command from `ClaimCenter\bin`:
   ```
   gwcc studio
   ```
4. In the Studio Resources pane, select **Localizations** → **Display Keys**.
5. Select the **Web.Login.Username** display key.
6. Change the value of the **Web.Login.Username** display key in an obvious manner. For example, add several question marks.
7. Save your changes in Studio.
8. Log into ClaimCenter with the `su` account. At this point the **User name** field is unchanged.
9. Press `ALT+SHIFT+T` to open the **Server Tools** page.
10. Select **Internal Tools.**
11. Select **Reload.**
12. Click **Reload PCF Files.**
13. Click **Log Out.** ClaimCenter redirects you to the login page.
14. Check for your change to the **User name** field. If you properly set up your development environment, the field now displays with the new name you set to the **Web.Login.Username** display key. You can revert this change once you have confirmed the successful configuration of your development environment.

After installing a ClaimCenter development environment with Tomcat, see the following topics for procedures you might want to use to set up your development environment:

- "Using the QuickStart Database" on page 37
- "Using SQL Server or Oracle in a Development Environment" on page 38
- "Installing Sample Data" on page 39
- "Configuring Logging" on page 35 in the *System Administration Guide*
- *ClaimCenter Configuration Guide*
- To integrate ClaimCenter with ContactCenter, see the *Guidewire Contact Management Guide*.

Using the QuickStart Database

By default, ClaimCenter uses the QuickStart database. The QuickStart database is the *H2 Database Engine*. Guidewire includes this database for the convenience of those who need a lightweight solution for demonstration and configuration purposes. By default, ClaimCenter creates and stores H2 database files within the `tmp\guidewire` directory of the local drive.

Setting the Database Configuration and Mode

Set configuration parameters for H2 in the `config.xml` file. For example:

```
<!-- H2 (meant for dev/quickstart use only!) -->
<database name="ClaimCenterDatabase" driver="dbcp"
        dbtype="h2" printcommands="false"
        autoupgrade="true" checker="false">
        <param name="jdbcURL" value="jdbc:h2:file:/tmp/guidewire/cc"/>
        <param name="CACHE_SIZE" value="32000"/></database>
```

Guidewire uses `\tmp\guidewire` as the database file location and `cc` as the file prefix in the default configuration. You can modify this location and prefix by modifying the value set to the `jdbcURL` parameter.

Limitations to the QuickStart Database

While the QuickStart database is convenient, Guidewire does not support using it for production. The following are limitations to using the QuickStart database:

- The QuickStart database only supports one connection at a time. Therefore, you cannot have the server running *and* look at the schema at the same time.
- You cannot test your cluster against the QuickStart database.
- The QuickStart database does not support all upgrades. Guidewire does not test upgrades on the QuickStart database other than the process of creating a database.

To drop your QuickStart database, open a command window at `ClaimCenter\bin` and enter `gwcc dev-dropdb`.

Additional References

- For more information about the QuickStart database and tools that you can download to view your schema, see `http://www.h2database.com`.
- For development environment information, see “The Studio Development Environment” on page 85 in the *Configuration Guide* and “Working with the QuickStart Development Server” on page 86 in the *Configuration Guide*.

Using SQL Server or Oracle in a Development Environment

ClaimCenter supports SQL Server and Oracle databases, as described in “Configuring the Database” on page 20. You can use Oracle or SQL Server instead of the QuickStart database. Such a configuration can be used for testing. Guidewire does not support this configuration for a production environment.

For instructions on creating a SQL Server or Oracle database instance, consult “Configuring the Database” on page 20. For instructions on configuring ClaimCenter to connect to the database, consult “Configuring a Database Connection” on page 44. After these configuration settings are complete, use the commands listed in “QuickStart Command Tools” on page 70 to control the application server.

Enabling Archiving for Development Testing

If you plan to use archiving in your production environment, enable archiving on a developer workstation so that you can design and test your custom implementation of the archiving plugin.

The default `config.xml` file has archiving disabled. This is set by the parameter:

```
<param name="ArchiveEnabled" value="false"/>
```

If you want to enable archiving, set the value of `ArchiveEnabled` to `true`. Then review the topics listed further in this section to learn how to configure archiving.

If you do not want to enable archiving, Guidewire recommends that you disable the archive work queue.

To disable the archive work queue

1. In a command window, navigate to the `ClaimCenter\bin` directory in the ClaimCenter installation.
2. Launch Guidewire Studio using the following command:
   ```
   gwcc studio
   ```
3. In Studio, open **Other Resources** → **work-queue.xml**.

4. Comment out the following block by adding `<!--` before the block and `-->` after it.

   ```
   <work-queue workQueueClass="com.guidewire.pl.domain.archiving.ArchiveWorkQueue"
    progressinterval="600000">
       <worker instances="1"/>
   </work-queue>
   ```

   If you have not previously edited `work-queue.xml`, click **Yes** when Studio prompts you to create a copy of `work-queue.xml` in the configuration module.

5. Select **File** → **Save Changes**.

See also

- "Archiving" on page 87 in the *Application Guide* – information on archiving claims, searching for archived claims, and restoring archived claims.
- "Archive Parameters" on page 39 in the *Configuration Guide* – discussion on the configuration parameters used in claims archiving.
- "Archiving Claims" on page 665 in the *Configuration Guide* – information on configuring claims archiving, selecting claims for archiving, and archiving and the object (domain) graph.
- "Archiving Integration" on page 285 in the *Integration Guide* – describes the archiving integration flow, storage and retrieval integration, and the `IArchiveSource` plugin interface.
- "Archive" on page 48 in the *Rules Guide* – information on base configuration archive rules and their use in detecting archive events and managing the claims archive and restore process.
- "Logging Successfully Archived Claims" on page 37 in the *System Administration Guide*.
- "Purging Unwanted Claims" on page 58 in the *System Administration Guide*.
- "Archive Info" on page 160 in the *System Administration Guide*.
- "Upgrading Archived Entities" on page 62 in the *Upgrade Guide*.

IMPORTANT To increase performance, most customers find increased hardware more cost effective than archiving unless their volume exceeds one million claims or more. Guidewire strongly recommends that you contact Customer Support before implementing archiving to help your company with this analysis.

Installing Sample Data

This topic explains how to load the sample data included with the base ClaimCenter installation, and describes how the sample data loading mechanism can be reconfigured.

ClaimCenter includes sample data for use in training, configuration and testing. Guidewire strongly recommends that you not load sample data into a production system.

To install sample data

1. Log into ClaimCenter with the `su` account.
2. Press `ALT+SHIFT+T` to open the **Server Tools** page.
3. Click **Internal Tools**.
4. Click **CC Sample Data** in the menu on the left.
5. Click **Load** for either the **Admin** or **Demo** datasets.

   ClaimCenter now contains some sample data.

About Importing and Exporting Data in ClaimCenter

For instructions on how to import or export administrative data, consult “Importing and Exporting Administrative Data” on page 115 in the *System Administration Guide*.

chapter 4

Installing a ClaimCenter Production Environment

Installing a ClaimCenter production environment is a multi-step process that requires you to perform several procedures. The initial installation process can take from two hours to a full day. This topic begins with an overview of the installation process and then guides you through each procedure culminating with the deployment of ClaimCenter to a production environment.

For instructions to install a ClaimCenter development environment, see "Installing a ClaimCenter Development Environment" on page 31 instead of this topic.

This topic includes:

Unpacking the Configuration Files

Guidewire packages ClaimCenter as a ZIP file. This file contains tools and files necessary to build a WAR or EAR file to install on an application server, and contains the developer toolkit and other items.

Unpack the configuration files onto the workstation that you plan to use as the home base for your ClaimCenter configuration. These directions assume you plan to maintain the configuration files on the administrative workstation.

To unpack the ClaimCenter configuration environment

1. If you have not already done so, create an installation directory for ClaimCenter. Do not use spaces in the installation directory path. This guide uses `ClaimCenter` as the directory name.
2. Unzip the ClaimCenter ZIP file into the ClaimCenter directory.
3. If you are not using a version control system, make a read-only copy of the ClaimCenter directory. This enables you to recover quickly from accidental changes that can prevent ClaimCenter from starting.
4. Open a command window to `ClaimCenter\bin`.
5. Enter the following command:
   ```
   gwcc copy-starter-resources
   ```
   This command copies configuration resources, including `config.xml`, to a configuration module at `ClaimCenter\modules\configuration`. Do not modify files in any module other than the `configuration` module.

At this point, you have a full set of ClaimCenter configuration files. For an overview of the directories included with ClaimCenter, see “ClaimCenter Configuration Files” on page 88 in the *Configuration Guide*.

Guidewire recommends that you maintain your ClaimCenter configuration files in a change control system such as Perforce or SVN. If you have such a system, add your ClaimCenter installation directory and files to it at this point. See “Linking Studio to a SCM System” on page 91 in the *Configuration Guide* for details.

Key ClaimCenter gwcc Commands

The following key commands are available with ClaimCenter after unpacking the ClaimCenter ZIP file. Launch these commands by passing the command name as a parameter to the `gwcc` utility in `ClaimCenter\bin`:

Command	Action
`gwcc -p`	Displays all `gwcc` command options.
`gwcc build-war`	Builds the generic WAR file for Tomcat. You can include the Boolean parameter `config.war.dictionary=true` to also generate the ClaimCenter *Data Dictionary* and *Security Dictionary* while building the WAR file. Use the following command: `gwcc build-war -Dconfig.war.dictionary=true` When `config.war.dictionary=true`, the command creates a `dictionary` folder within the WAR file. The `dictionary` folder contains `data` and `security` folders. These folders contain the *Data Dictionary* and *Security Dictionary* respectively. To view a dictionary, open `index.html` in the `data` or `security` folder.
`gwcc build-weblogic-ear`	Builds the EAR file for WebLogic.
`gwcc build-websphere-ear`	Builds the EAR file for WebSphere.
`gwcc copy-starter-resources`	Copies starter configuration resources to the configuration module, if applicable. Class: `com.guidewire.tools.config.CopyStarterConfigResourcesTool`
`gwcc regen-datamapping-split`	Builds the data mapping files with files split out by table and typelist. Class: `com.guidewire.tools.datamapping.DataMappingTool`
`gwcc regen-datamapping-together`	Builds the data mapping files with all tables and typelists concatenated. Class: `com.guidewire.tools.datamapping.DataMappingTool`
`gwcc regen-dictionary`	Generates the *ClaimCenter Data Dictionary* and *ClaimCenter Security Dictionary*. Generate the first time you unzip the application and each time you update the data model. Run the `gwcc regen-java-api` and `gwcc regen-soap-api` commands each time just prior to regenerating the security and data dictionaries. To view either, open a new browser window and enter: • `ClaimCenter/dictionary/data/index.html` for the *Data Dictionary* or • `ClaimCenter/dictionary/security/index.html` for the *Security Dictionary.* You can also generate these dictionaries while building a WAR file. See the description for `gwcc build-war` for instructions. Classes: `com.guidewire.tools.dictionary.data.DataDictionaryTool` `com.guidewire.tools.dictionary.security.SecurityDictionaryTool`
`gwcc regen-gosudoc`	Generates Gosu documentation similar to JavaDoc but using the ClaimCenter type system. This command produces documentation at `ClaimCenter/build/gosudoc/index.html`. See "Gosu Generated Documentation" on page 35 in the *Gosu Reference Guide*. Class: `com.guidewire.tools.gosudoc.GosuDocMain`
`gwcc regen-java-api`	Builds the Java API toolkit and examples to the `ClaimCenter/java-api` directory. See "Regenerating the Integration Libraries" on page 17 in the *Integration Guide*. Class: `com.guidewire.commons.entity.external.ExternalGenerator`
`gwcc regen-pcfmapping`	Builds the PCF mappings. Class: `com.guidewire.tools.pcfmapping.PCFMappingWriterMain`
`gwcc regen-rulereport`	Generates an XML report describing the existing business rules. See "Generating a Rule Repository Report" on page 91 in the *Rules Guide*.

Command	Action
`gwcc regen-soap-api`	Builds the SOAP API toolkit and examples to the `ClaimCenter/soap-api` directory. See “Regenerating the Integration Libraries” on page 17 in the *Integration Guide*. Classes: `com.guidewire.tools.wsdl.WSDLGenerator` `com.guidewire.util.webservices.axis.WSDLToJavaGenerator`
`gwcc regen-xsd`	Builds the XSD files for data import. See “Creating an XML File for Import” on page 121 in the *System Administration Guide* and “Importing Administrative Data” on page 83 in the *Integration Guide*.
`gwcc studio`	Runs Guidewire Studio. Class: `com.guidewire.studio.main.Main`
`gwcc verify-checksum`	Verifies module checksums. Class: `com.guidewire.tools.checksum.ModulesChecksumTool`
`gwcc verify-types`	Checks PCF files for errors. See“Validating Studio Resources” on page 121 in the *Configuration Guide*.
`gwcc version`	Displays the product version.

This list does not include commands specific to starting and stopping the QuickStart server. For a full list of `gwcc` commands, see “Commands Reference” on page 69.

Configuring a Database Connection

You set database connections by uncommenting the appropriate sample `database` element in the `config.xml` file, accessible from the Guidewire Studio **Resources** pane under **configuration** → **Other Resources**. In this file you supply connection and configuration parameters for your database.

This topic includes the following:

- “The database Element” on page 44
- “Defining the jdbcURL” on page 46
- “Specifying a Database Password” on page 47
- “Enabling SQL Server JDBC Logging” on page 48
- “Configuring ClaimCenter to Use a JNDI Data Source” on page 48
- “Creating a JNDI Data Source on WebLogic” on page 49
- “Creating a JNDI Data Source on WebSphere” on page 51

After you have finished configuring the database connection, proceed to “Deploying ClaimCenter to the Application Server” on page 56 for instructions on deploying ClaimCenter to the application server.

The database Element

The `database` element in `config.xml` has the following structure:

```
<database name="string" driver="dbcp|jndi" env="string" dbtype="type" autoupgrade="boolean"
  verifyschema="boolean" checker="boolean" addforeignkeys="boolean" passwordFile="string"
  characterset="string" automaticallycreatetablespaces="boolean">
  <tablespacemapping logicalname="string" physicalname="string"/>
  <param name="string" value="string"/>
  <databasestatistics>
  <!-- See “Configuring Database Statistics” on page 53 in the System Administration Guide
            for an explanation of these elements. -->
  </databasestatistics>
</database>
```

The attributes in the `database` element are:

Attribute	Description
`name`	A string specifying the database name.
`driver`	The driver used to connect to the database. If you want ClaimCenter to manage the database connection, leave `driver` set to `dbcp`. If you want to use a JNDI datasource managed by the application server, set `driver` to `jndi` and see "Using a JNDI Data Source" on page 48.
`env`	A string identifying the environment in which ClaimCenter uses this connection specification. See "env Property" on page 20 in the *System Administration Guide* for information about using this element.
`dbtype`	The database vendor. Either `h2` (for the QuickStart database), `oracle` or `sqlserver`.
`autoupgrade`	A Boolean value specifying whether the server automatically upgrades the database as it starts. The default is false. This parameter is optional. See "Understanding and Authorizing Database Upgrades" on page 50 in the *System Administration Guide*.
`printcommands`	A Boolean value specifying whether the server prints database upgrade messages to the console upon startup. By default, `printcommands` is set to `true`. Do not set `printcommands` to `false` in a production environment.
`checker`	A Boolean value specifying whether ClaimCenter runs consistency checks before it starts. For development environments with small data sets, you can enable consistency checks to run each time the ClaimCenter server starts. Set the `checker` attribute of the database block to `true` to enable to enable checks on startup. By default, this option is set to `false`. Running consistency checks upon starting the server can take a long time, impact performance severely, and possibly time out on very large datasets. Set `checker` to `false` under most circumstances. Guidewire recommends that you do not set `checker` to `true` except in development environments with small test data sets. This parameter is optional. See "Checking Database Consistency" on page 51 in the *System Administration Guide*.
`addforeignkeys`	Used only for development and testing. Do not use this attribute in production.
`verifyschema`	A Boolean value specifying whether ClaimCenter verifies the database schema against the data model files following an upgrade. The `verifyschema` attribute only applies if `autoupgrade` is set to `true`.
`automaticallycreatetablespaces`	Used only for development and testing. Do not use this attribute in production.
`passwordFile`	Specifies a file containing the database password. This parameter is optional. See "Specifying a Database Password" on page 47 for more information.
`characterset`	For SQL Server, this specifies the Windows or SQL collation name. ClaimCenter ignores this parameter if specified for other databases. Only use this parameter while creating a new database in a development environment. See your SQL Server documentation for information about collation names.

Mapping Logical Tablespaces to Physical Tablespaces

The `database` element has a `tablespacemapping` subelement that maps the logical tablespaces required by ClaimCenter to either Oracle tablespaces or SQL Server filegroups. For Oracle, mapping to tablespaces is required. For SQL Server, mapping to filegroups is optional. Create these tablespaces or filegroups when you set up your database. See "Configuring the Database" on page 20 for information on creating tablespaces or filegroups for your database.

Defining Table Groups

You can define zero or more table groups within the `database` element. You can use table groups to specify a set of tables on which to run database consistency checks.

To define a table group, add a `tablegroup` element to the `database` element in `config.xml`. The `tablegroup` element has a `name` attribute and a `tables` attribute. The `name` attribute identifies the table group. The `tables` attribute defines which tables are in the table group. Tables are listed in a comma-separated list. For example:

```
<database>
...
  <tablegroup name="MyTables" tables="cc_sometable1, cc_someTable2, cc_someTable3"/>
...
</database>
```

When the ClaimCenter server starts, it checks that no table is listed more than once in a single table group definitions and that each table listed exists. If a table is listed more than once in a group or does not exist, the ClaimCenter server logs an error and stops.

For instructions to run consistency checks from the ClaimCenter application for a particular table group, see "Consistency Checks" on page 163 in the *System Administration Guide*. For instructions to run consistency checks from the command line, see "system_tools Command" on page 173 in the *System Administration Guide*.

Specifying Additional Database Parameters

To pass additional parameters required for your database connection, specify one or more name-value `param` elements. Several parameters configure the connection pool, if you are using ClaimCenter to manage the connection pool rather than a JNDI data source managed by the application server. Parameters that control the connection pool are described in "Configuring Connection Pool Parameters" on page 47 in the *System Administration Guide*.

Defining the jdbcURL

The `jdbcURL` parameter stores connection information for the database. Define a `jdbcURL` parameter unless you use a JNDI data source managed by WebLogic or WebSphere. If you want to use a JNDI data source managed by WebLogic or WebSphere, skip to "Configuring ClaimCenter to Use a JNDI Data Source" on page 48.

Defining jdbcURL for Oracle

Specify the `jdbcURL` for a standalone Oracle instance in either of the following formats:

```
<param name="jdbcURL" value="jdbc:oracle:thin:userName/password@serverName:port:OracleSID"/>
```

or

```
<param name="jdbcURL" value="jdbc:oracle:thin:userName/password@
 (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=serverName)(PORT=port))
 (CONNECT_DATA=(SERVICE_NAME=OracleSID)))"/>
```

For Oracle Real Application Cluster (RAC) implementations, use an expanded form of the latter `jdbcURL` format. For example, a two node Oracle cluster

```
<param name="jdbcURL" value="jdbc:oracle:thin:userName/password@
 (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=node1)(PORT=port))
 (ADDRESS=(PROTOCOL=TCP)(HOST=node2)(PORT=port))(LOAD_BALANCE=yes)
 (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=OracleSID)))"/>
```

The default port number for Oracle is 1521.

You can specify the *`serverName`* as the computer name or IP address.

Guidewire bundles the Oracle 11.1.0.7 Production JDBC Thin Driver, pure Java, Type IV driver in `ClaimCenter\admin\lib\ojdbc6.jar`. Oracle documentation describes how to format the URL. Refer to the following page for more information:

```
http://download.oracle.com/docs/cd/B19306_01/java.102/b14355/urls.htm#BEIJFHHB
```

Defining jdbcURL for SQL Server

Specify the `jdbcURL` for SQL Server in the following format:

```
<param name="jdbcURL" value="jdbc:sqlserver://serverName[:port];
 databaseName=cc;user=ccUser;password=password
 [;applicationName=applicationName]"/>
```

Optional parameters are shown in brackets.

The default port number for SQL Server is 1433. If your SQL Server instance listens on the default port, 1433, you can omit the port and preceding colon from the `jdbcURL`. However, Microsoft recommends that you always specify the port value for security reasons. When you specify the port number, the JDBC driver connects directly to SQL Server and does not make a request to sqlbrowser. If your SQL Server instance is listening to a different port than the default 1433, specify that port number in the `jdbcURL`.

You can include an `applicationName` property on the `jdbcURL` connection string. If you set this value, the server logs and **Activity Monitor** include it when identifying threads. If you do not specify an `applicationName`, the ClaimCenter server creates one by concatenating `cc`, followed by the application version, including build number. Finally, if you defined the SQL Server `ADDL_CONN_DESCR` system property, ClaimCenter appends this value to the generated `applicationName` value.

ClaimCenter requires that the `selectMethod` on the `jdbcURL` connection be set to `direct`. This is the default value, so you do not need to include this value in your `jdbcURL`. If you include `selectMethod` and set it to `cursor`, the server does not start.

ClaimCenter defaults the value of the `sendStringParametersAsUnicode` property to be the correct, appropriate value in the SQL Server `jdbcURL` connection. ClaimCenter does not override an existing value. If you set `sendStringParametersAsUnicode` in `config.xml`, the server will validate the value to be correct.

Specifying a Database Password

Supply a password with your database connection parameters. To store the database password in the `config.xml` file, set the password in the `jdbcURL` parameter of the `<database>` block in `config.xml`. If you do not want to expose this password in `config.xml`, Guidewire provides you with the following alternatives:

- “Using a Password File” on page 47
- “Using the Database Authentication Plugin” on page 47
- “Using a JNDI Data Source” on page 48

Using a Password File

You can place the password in an external file and reference this file from the `<database>` block of the `config.xml` file.

To use a password file

1. Add the `passwordFile` attribute to the `<database>` element.
2. Set the value of `passwordFile` to the absolute path of the password file.
3. Replace the `Password` value in the connection specification with a `{$password}` placeholder.

   At run time, ClaimCenter reads the password from the file.

   When you are done, your database specification looks similar to the following:

```
<database name="ClaimCenterDatabase" driver="dbcp"
 dbtype="sqlserver" autoupgrade="true" checker="false"
 passwordFile=”c:\secure\password.txt”>
  <param name="jdbcURL" value="jdbc:sqlserver://HOSTNAME:1433;
   databaseName=cc;user=ccUser;
   password=${password}" />
  <param name="maxWait" value="30000" />
  <param name="checker.threads" value="1" />
</database>
```

Using the Database Authentication Plugin

For an even higher level of security, Guidewire provides a database authentication plugin: `DBAuthenticationPlugin`. You can use this plugin to define a custom method that returns the user name and password in a format that the database system recognizes. For information on implementing this in your environment, see “Database Authentication Plugins” on page 245 in the *Integration Guide*.

Using a JNDI Data Source

You can specify for ClaimCenter to use a JNDI data source configured through a WebLogic or WebSphere application server for your database connection. This data source uses a Java 2 Connector (J2C) authentication alias to store the user name and password. See “Configuring ClaimCenter to Use a JNDI Data Source” on page 48 for details.

Enabling SQL Server JDBC Logging

During troubleshooting, Guidewire might request a trace log from the Microsoft JDBC driver. This topic describes how to enable trace logging for SQL Server. Do not turn on logging in other circumstances as it places a heavy overhead on the system, and the files created can quickly become very large.

Microsoft JDBC driver logging can be turned on at startup for ClaimCenter by specifying two database parameters:

```
<database ...>
  ...
  <param name="msjdbctracelevel" value="FINER"/>
  <param name="msjdbctracefile" value="c:/temp/info.log"/>
</database>
```

The trace level is a string that corresponds to a valid trace level as documented at `http://msdn.microsoft.com/en-us/library/ms378517(SQL.90).aspx?ppud=4`. The trace file can be specified, or defaults to `C:\temp\msjdbctrace%u.log`. The trace file specified is a pattern documented at `http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/FileHandler.html`. Using `%h` and `%t` puts the file in the `Documents and Settings` directory under the name which is running the application server.

A page called **Microsoft JDBC Driver Logging** is available in the ClaimCenter **Info Pages**. This page enables you to start and stop Microsoft driver logging on a running application server. Using this page might be a better option when tracing a particular operation, in order to minimize system impact and size of the trace file. To turn tracing on, choose a logging level, simple or XML logging format and a log file location. Click **Set Logging Level**. Messages report the outcome of the operation.

If logging has already been enabled through `config.xml` or previous use of this page, then the logging level resets to the new level. ClaimCenter flushes and closes any existing logging files before beginning the new trace. If `OFF` is the chosen logging level, logging is turned off. Any existing logging files are flushed and closed.

The ability to control logging of the Microsoft JDBC driver through ClaimCenter only works when using the internal connection pool, not when using an external WebSphere or WebLogic connection pool.

Configuring ClaimCenter to Use a JNDI Data Source

ClaimCenter can use a Java Naming and Directory Interface (JNDI) data source managed by a WebLogic or WebSphere application server. This enables you to configure database parameters, including connection pool size, using the application server. Using a JNDI data source also provides you with another secure alternative to placing the user name and password in the `config.xml` file.

IMPORTANT Guidewire only supports JNDI using the drivers bundled with ClaimCenter. Guidewire does not support the XA versions of a data source.

By default, the database connection in the `config.xml` defines a Java Database Connectivity (JDBC) specification for connecting to the database. To access this connection through JNDI, remove the JDBC parameters in your `database` element and replace them with JNDI parameters.

To configure ClaimCenter to use a JNDI data source

1. Open `config.xml` from the Guidewire Studio **Resources** pane under **configuration** → **Other Resources**.
2. Set the `driver` attribute to `jndi`.
3. Remove the `jdbcURL` parameter.
4. Add a `jndi.datasource.name` parameter and specify the JNDI name you assign to the data source.

   When you are finished, the `database` element looks similar to the following:

   ```
   <database name="ClaimCenterDatabase" driver="jndi" dbtype="oracle|sqlserver"
     autoupgrade="false">
     <param name="jndi.datasource.name" value="jdbc/ccDataSource" />
     ...
    </database>
   ```
5. Close and save the `config.xml` file.
6. Rebuild and install the ClaimCenter application EAR file. See "Deploying ClaimCenter to the Application Server" on page 56 for instructions.

Before deploying ClaimCenter to the application server, create the JNDI data source on the application server. See one of the following topics, depending on your application server type:

- "Creating a JNDI Data Source on WebLogic" on page 49
- "Creating a JNDI Data Source on WebSphere" on page 51

During startup, ClaimCenter records the connection made through JNDI with an entry similar to the following in the log:

```
2008-06-21 10:49:13,260 INFO Looking up JNDI datasource 'jdbc/ccDataSource'...
```

Creating a JNDI Data Source on WebLogic

This topic describes how to create a JNDI data source on WebLogic. To configure ClaimCenter to use the data source, see "Configuring ClaimCenter to Use a JNDI Data Source" on page 48.

This topic includes the following:

- "Creating an Oracle JNDI Data Source on WebLogic" on page 49
- "Creating a SQL Server JNDI Data Source on WebLogic" on page 50

Creating an Oracle JNDI Data Source on WebLogic

This section describes how to create an Oracle JNDI data source on WebLogic.

To verify the Oracle driver used by WebLogic matches the one bundled with ClaimCenter

1. Unpack the `ojdbc6.jar` files in the WebLogic `server\lib` directory and the `ClaimCenter\admin\lib` directory.
2. Open the `MANIFEST.MF` file of each `ojdbc6.jar` file in a text editor. The property values in both must match identically. The Oracle driver bundled with ClaimCenter is the same as the Oracle driver bundled with the supported WebLogic version.

If the values in the manifest files do not match exactly, do one of the following. Copy the `ojdbc6.jar` file from `ClaimCenter\admin\lib` to the WebLogic `server\lib` directory. Or, open the `setDomainEnv` file, and prepend the absolute path to `ClaimCenter\admin\lib\ojdbc6.jar` to the `PRE_CLASSPATH` environment variable. The WebLogic server must be restarted for this change to take effect.

To create the data source

1. Open the WebLogic Server Administration Console.
2. Choose **Services** → **JDBC** → **Data Sources**.

3. Click **New**.
4. Enter a name and JNDI name for the data source. The JNDI name must match the value of the `jndi.datasource.name` parameter in the `config.xml` file.
5. Select **Oracle** as the **Database Type**.
6. Select **Oracle's Driver (Thin) Versions:9.0.1,9.2.0,10,11** as the **Database Driver** and click **Next**.
7. Uncheck **Supports Global Transactions** and click **Next**.
8. Fill in connection properties for your environment and click **Next**. The WebLogic Server Administration Console displays the **Test Database Connection** page.
9. Set the **Driver Class Name** to `oracle.jdbc.pool.OracleDataSource`.
10. Click **Test Configuration**. If you configured the connection properly and the database is running, WebLogic displays the message "Connection test succeeded."
11. Click **Next**.
12. Select the servers that use the data source.
13. Click **Finish**. The WebLogic Server Administration Console returns you to the **Summary of JDBC Data Sources** page.
14. Click **Activate Changes**. WebLogic displays the message "All changes have been activated. No restarts are necessary."

Creating a SQL Server JNDI Data Source on WebLogic

This section describes how to create a SQL Server JNDI data source on WebLogic.

To copy the sqljdbc4.jar file to WebLogic

1. Copy the `sqljdbc4.jar` file from the `ClaimCenter\admin\lib` directory to the `server\lib` directory within the WebLogic home. The `sqljdbc4.jar` file contains the APIs for connecting to the ClaimCenter database. This file must be in the classpath of the WebLogic domain.
2. Add the `sqljdbc4.jar` file to the classpath of the WebLogic domain.

   To modify the classpath for a single domain, open the *`WL_HOME`*`/common/bin/commEnv` script appropriate to your operating system in a text editor. Then, prepend the absolute path to the `sqljdbc4.jar` file, including file name, to the `WEBLOGIC_CLASSPATH` environment variable. The WebLogic server must be restarted for this change to take effect.

   To modify the classpath for all domains, open the `setDomainEnv` script appropriate to your operating system in a text editor. Then, prepend the absolute path to the `sqljdbc4.jar` file to the `PRE_CLASSPATH` environment variable. The WebLogic server must be restarted for this change to take effect.

To create the data source

1. Open the WebLogic Server Administration Console.
2. Click **Services** → **JDBC** → **Data Sources**.
3. Click **New**.
4. Enter a **Name** and **JNDI Name** for the data source. The JNDI name must match the value of the `jndi.datasource.name` parameter in the `config.xml` file.
5. Select **MS SQL Server** as the **Database Type**.
6. Select **Other** as the **Database Driver**, and click **Next**.
7. Uncheck **Supports Global Transactions**, and click **Next**.

8. Specify connection properties for the SQL Server database, and click **Next**.
9. Enter `com.microsoft.sqlserver.jdbc.SQLServerDriver` for the **Driver Class Name**.
10. Specify the **URL** using the following format:

    ```
    jdbc:sqlserver://servername:port;databasename=dbname;user=username
    ```
11. Fill in connection properties for your environment. If using a unicode database, set the property `sendStringParametersAsUnicode` to `true`. If using a single-byte database, set `sendStringParametersAsUnicode` to `false`.
12. Click **Test Configuration**. If you configured the connection properly and the database is running, WebLogic displays the message "Connection test succeeded."
13. Click **Next**.
14. Select the targets that use the data source.
15. Click **Finish**. The WebLogic Server Administration Console returns you to the **Summary of JDBC Data Sources** page.
16. Click **Activate Changes**. WebLogic displays the message "All changes have been activated. No restarts are necessary."

Creating a JNDI Data Source on WebSphere

This topic describes how to create a JNDI data source on WebSphere. To configure ClaimCenter to use the data source, see "Configuring ClaimCenter to Use a JNDI Data Source" on page 48.

This topic includes the following:
- "Creating an Oracle JNDI Data Source on WebSphere" on page 51
- "Creating a SQL Server JNDI Data Source on WebSphere" on page 53

Creating an Oracle JNDI Data Source on WebSphere

This section describes how to create an Oracle data source on WebSphere.

To create the JDBC provider

1. Open the WebSphere Administrative Console if not already open.
2. Choose **Resources** → **JDBC** → **JDBC Providers**.
3. Set the **Scope** to **Cell**.
4. Click **New** to create a new JDBC provider.
5. Select **Oracle** for the **Database type**.
6. Select **Oracle JDBC Driver** for the **Provider type** drop down and click **OK**.
7. Select **Connection pool data source** for the **Implementation type**.
8. Supply a new **Name** for the JDBC provider, for example `ccOracle`.
9. Enter a **Description** for the JDBC provider if you want.
10. Click **Next**.
11. Specify the directory location of `ojdbc6.jar`. Set the value to the `WEB_INF\lib` path within the installed ClaimCenter instance on WebSphere. For example, you might set it to:

    ```
    WAS HOME\AppServer\profiles\profile\installedApps\cell\cc.ear\cc.war\WEB-INF\lib
    ```
12. Click **Next**.

13. Review the **Summary** page. Click **Previous** if you need to make changes. Otherwise, click **Finish**.
14. Click **Save** to apply your changes to the master configuration.

    WebSphere returns you to the **JDBC Providers** page. At this point, you have completed the creation of the new provider.

To create the data source

1. Open the WebSphere Administrative Console if not already open.
2. Choose **Resources** → **JDBC** → **JDBC Providers**.
3. Set the **Scope** to **Cell**.
4. Select the JDBC provider that you created for Oracle.

   WebSphere displays the **Configuration** tab.
5. Select **Data Sources** in the **Additional Properties** section.

   WebSphere displays the **Data sources** page.
6. Click **New** to create a new data source.
7. Enter a **JNDI name** for the data source. This value must match the value set to `jndi.datasource.name in config.xml.` See "Configuring ClaimCenter to Use a JNDI Data Source" on page 48.
8. Click **Next**.
9. Set the **URL** value using the `jdbc:oracle:thin:@`*`hostname`*`:`*`port`*`:`*`ORACLE_SID`* format.
10. Select **Oracle11g data store helper** for the **Data store helper class name**.
11. Click **Next**.
12. If you do not yet have a J2C (Java 2 Connector) authentication alias, create a new one.

    a. Click **Global J2C authentication alias**.

    b. Click **New**.

    c. Enter the following.

Parameter	Value
`Alias`	A string specifying the alias name. this can be anything you like, for example `ccAlias`.
`User ID`	A string specifying the user name.
`Password`	A string specifying the password.

    d. Click **OK**.

    e. Click **Save** to save apply your changes to the master configuration.

    f. Start the procedure to create the data source again, beginning with step 2.
13. Select an authentication alias for the **Component-managed authentication alias**.
14. Select an authentication alias for the **Container-managed authentication alias**.
15. Click **Next**. Review the information on the **Summary** page. Click **Previous** if you need to make changes. Otherwise, click **Finish**.
16. Click **Save** to save apply your changes to the master configuration.

To configure data source properties

1. Open the WebSphere Administrative Console if it is not already open.

2. Choose **Resources** → **JDBC** → **Data sources**.
3. Click the data source that you just defined. WebSphere displays the **Configuration** tab.
4. Click **WebSphere Application Server data source properties**.
5. Set **Statement cache size** to `0`.
6. Select **Non-transactional data source**.
7. Click OK.
8. Under **Additional Properties**, click **Custom Properties**.
9. At the top of the **Custom properties** page, click **New.**
10. Enter the **Name** `commitOrRollbackOnCleanup`.
11. Enter the **Value** `rollback`.
12. Click **OK**.
13. Click **Save** to apply your changes to the master configuration.

To test the connection

1. In the WebSphere Administrative Console, select **Resources** → **JDBC** → **Data sources** to return to the **Data sources** page.
2. Select the checkbox next to your ClaimCenter data source.
3. Click **Test Connection** to verify that the connection works.

Creating a SQL Server JNDI Data Source on WebSphere

This section describes how to create a SQL Server JNDI data source on WebSphere.

To copy the sqljdbc4.jar file to WebSphere

Before you begin, copy the `sqljdbc4.jar` file from `ClaimCenter\admin\lib` to the WebSphere `lib\ext` directory. The `sqljdbc4.jar` file contains the APIs for connecting to the ClaimCenter database.

To create the JDBC provider

1. Open the WebSphere Administrative Console if not already open.
2. Choose **Resources** → **JDBC** → **JDBC Providers**.
3. Set the **Scope** to **Cell**.
4. Click **New** to create a new JDBC provider.
5. Select **SQL Server** for the **Database type**.
6. Select **Microsoft SQL Server JDBC Driver** for the **Provider type** drop down and click **OK**.
7. Select **Connection pool data source** for the **Implementation type**.
8. Supply a new **Name** for the JDBC provider, for example `ccSQLServer`.
9. Enter a **Description** for the JDBC provider if you want.
10. Click **Next**.
11. Specify the directory location of `sqljdbc4.jar`. Set the value to the WebSphere `lib\ext` directory. For example, you might set it to:

    ```
    C:\Program Files\IBM\WebSphere\AppServer\lib\ext
    ```

You can ignore that the greyed out **Class path** box lists the JAR name as `sqljdbc.jar` instead of `sqljdbc4.jar`.

You do not need to enter a value for the **Native library path**.

12. Click **Next**.
13. Review the **Summary** page. Click **Previous** if you need to make changes. Otherwise, click **Finish**.
14. Click **Save** to apply your changes to the master configuration.

    WebSphere returns you to the **JDBC Providers** page. At this point, you have completed the creation of the new provider.
15. Select the JDBC provider you just created.
16. Edit the **Class path** to change `${MICROSOFT_JDBC_DRIVER_PATH}/sqljdbc.jar` to `${MICROSOFT_JDBC_DRIVER_PATH}/sqljdbc4.jar`.
17. Click **OK**.
18. Click **Save** to apply your changes to the master configuration.

To create the data source

1. Open the WebSphere Administrative Console if it is not already open.
2. Choose **Resources** → **JDBC** → **JDBC Providers**.
3. Set the **Scope** to **Cell**.
4. Select the JDBC provider that you created for SQL Server.

   WebSphere displays the **Configuration** tab.
5. Select **Data Sources** in the **Additional Properties** section.

   WebSphere displays the **Data sources** page.
6. Click **New** to create a new data source.

   Enter a **JNDI name** for the data source, such as `jdbc/ccDataSource`. This value must match the value set to `jndi.datasource.name in config.xml.` See “Configuring ClaimCenter to Use a JNDI Data Source” on page 48.
7. Click **Next**.
8. Enter the **Database name**.
9. If using a different port to connect to the database than the default of 1433, change the **Port number** value.
10. Enter the **Server name** for the server hosting SQL Server.
11. Uncheck the box for **Use this data source in container managed persistence (CMP)**.
12. Click **Next**.
13. If you do not yet have a J2C (Java 2 Connector) authentication alias, create a new one.
    a. Click **Global J2C authentication alias**.
    b. Click **New**.
    c. Enter the following.

Parameter	Value
`Alias`	A string specifying the alias name. this can be anything you like, for example `ccAlias`.
`User ID`	A string specifying the user name.

Parameter	Value
`Password`	A string specifying the password.

d. Click **OK**.

e. Click **Save** to save apply your changes to the master configuration.

f. Start the procedure to create the data source again, beginning with step 2.

14. Select an authentication alias for the **Component-managed authentication alias**.
15. Select an authentication alias for the **Container-managed authentication alias**.
16. Click **Next**. Review the information on the **Summary** page. Click **Previous** if you need to make changes. Otherwise, click **Finish**.
17. Click **Save** to save apply your changes to the master configuration.

To configure data source properties

1. Open the WebSphere Administrative Console if it is not already open.
2. Choose **Resources** → **JDBC** → **Data sources**.
3. Click the data source you created for SQL Server.
4. Under **Additional Properties**, click **WebSphere Application Server data source properties**.
5. Set the **Statement cache size** to `0`.
6. Select the **Non-transactional data source** checkbox.
7. Click **OK**.
8. Click **Save** to apply your changes to the master configuration.
9. Click the data source you created for SQL Server.
10. Under **Additional Properties**, click **Custom Properties**.
11. Check the property **sendStringParametersAsUnicode**. If you are using unicode columns (`nvarchar`), then this must be set to `true`. If you are not using `nvarchar`, but single byte `varchar` columns, this must be set to `false`. Your application server will not start up if this setting is incorrect. If you need to change the value, click the property name.
12. At the top of the **Custom properties** page, click **New**.
13. Enter the **Name** `commitOrRollbackOnCleanup`.
14. Enter the **Value** `rollback`.
15. Click **OK**.
16. Click **Save** to apply your changes to the master configuration.

To test the connection

1. In the WebSphere Administrative Console, select **Resources** → **JDBC** → **Data sources** to return to the **Data sources** page.
2. Select the checkbox next to your ClaimCenter data source.
3. Click **Test Connection** to verify that the connection works.

Deploying ClaimCenter to the Application Server

After you have defined the database connection specification, deploy ClaimCenter to the application server. Deploying ClaimCenter requires creating a WAR or EAR file and installing the package on an application server.

This topic includes the following:

- "Installing a JBoss Production Environment" on page 56
- "Installing a Tomcat Production Environment" on page 57
- "Installing a WebLogic Production Environment" on page 58
- "Installing a WebSphere Production Environment" on page 59

These instructions are for creating a production ClaimCenter environment. For instructions to create a development ClaimCenter environment, see "Installing a ClaimCenter Development Environment" on page 31

Installing a JBoss Production Environment

If you are running JBoss in a 64-bit JVM, you might need to increase the heap size to prevent JBoss from running out of memory. You can set Java options in the JBoss `bin\run.bat` or `bin\run.sh` file by adding a line similar to the following:

```
set JAVA_OPTS=-Xms512M -Xmx1024M -XX:MaxPermSize=512M -Dsun.rmi.dgc.client.gcInterval=3600000
-Dsun.rmi.dgc.server.gcInterval=3600000 -Dorg.jboss.resolver.warning=true -server
```

Alternatively, you can specify the Java options by creating a `JAVA_OPTS` environment variable.

The specific values used for `JAVA_OPTS` in this example might not be suitable for your environment. Contact Guidewire if you need assistance.

To deploy ClaimCenter to JBoss, remove conflicting libraries, add any additional servlets, then generate and deploy the WAR file.

To remove conflicting libraries

1. Remove the `lib/endorsed/stax-api.jar` file from your JBoss installation to prevent class loader conflicts with ClaimCenter.
2. Remove the `lib/endorsed/xercesImpl.jar` file from your JBoss installation to avoid SAX parser exceptions.

To add servlet definitions

1. Copy the `ClaimCenter\modules\cc\deploy\WEB-INF\web.xml` file to `ClaimCenter\modules\configuration\deploy\WEB-INF\web.xml`.
2. Open `web.xml` in a text editor.
3. Add `servlet` definitions as needed. See the defined servlets for an example.
4. Add a `servlet-mapping` definition for each servlet that you add. See the defined servlet mappings for an example.
5. Save `web.xml`.

To generate and deploy the JBoss WAR file

1. Open a command prompt.
2. Navigate to `ClaimCenter/bin`.
3. Run the following command:
   ```
   gwcc build-jboss-war
   ```

This command creates a `cc.war` file and places it in the `ClaimCenter/dist/war` directory.

4. Copy `cc.war` to the `deploy` directory in a server configuration directory, for example, *`JBOSS_DIST/`*`jboss-as/server/default/deploy`. This example uses the default server configuration directory.
5. Copy the appropriate database driver JAR file from `ClaimCenter/admin/lib` to *`JBOSS_DIST/`*`jboss-as/server/default/lib`. Copy the JAR file that contains database drivers for your database type:

Database	Driver JAR
Oracle	`ojdbc6.jar`
SQL Server	`sqljdbc4.jar`

Installing a Tomcat Production Environment

Tomcat requires a WAR package file. Before you deploy to Tomcat, check that you have defined the `CATALINA_OPTS` variable with a value of:

```
-Xms1024M -Xmx1024M -XX:PermSize=128m -XX:MaxPermSize=128M
```

Securing Application Resources with J2EE Security Constraints

By default, all users are able to browse all content within a particular web application. This includes documents that can contain sensitive information, such as `config.xml`. The Java servlet specification defines how to configure security constraints to prevent users from browsing these files.

The ClaimCenter `WEB-INF\web.xml` file must be modified to configure the security constraints. This file contains information about which application resources to restrict and how to restrict them. You can also choose to define additional servlets in `web.xml`.

The minimal amount of configuration required to prevent anyone from accessing a file such as `config.xml` is to add a security constraint within the `<web-app>` tag inside `web.xml`:

To add the security constraint

1. Copy the `ClaimCenter\modules\cc\deploy\WEB-INF\web.xml` file to `ClaimCenter\modules\configuration\deploy\WEB-INF\web.xml`.
2. Open `web.xml` in a text editor.
3. Add the following within the `<web-app>` tag:

```
<security-constraint>
  <display-name>Tomcat Server Configuration Security Constraint</display-name>
  <web-resource-collection>
    <web-resource-name>Protected Area</web-resource-name>
    <!-- Define the context-relative URL(s) to be protected -->
    <url-pattern>/config/config.xml</url-pattern>
    <http-method>POST</http-method>
    <http-method>GET</http-method>
  </web-resource-collection>
  <auth-constraint>
    <!-- Anyone with one of the listed roles may access this area -->
    <role-name>admin</role-name>
  </auth-constraint>
</security-constraint>
```

   Some further configuration is shown inside the `web.xml` that routes people to an error page.

4. Save `web.xml`.

Deploying the Tomcat WAR File

Define any necessary servlets in `web.xml`. Then build a WAR file and deploy the WAR file to Tomcat.

To add servlet definitions

1. Open `ClaimCenter\modules\configuration\deploy\WEB-INF\web.xml`.
2. Add `servlet` definitions as needed. See the defined servlets for an example.
3. Add a `servlet-mapping` definition for each servlet that you add. See the defined servlet mappings for an example.
4. Save `web.xml`.

To deploy to Tomcat

1. Open a command window.
2. Navigate to `ClaimCenter\bin`.
3. Run the following command:
   ```
   gwcc build-tomcat-war
   ```
   This command creates a `cc.war` file and places it in the `ClaimCenter\dist\war` directory.
4. Deploy the package to Tomcat by copying the `cc.war` file to the `webapps` directory in your Tomcat server.

   When Tomcat starts up, it automatically recognizes ClaimCenter and unpacks the `cc.war` into a directory structure within `Tomcat\webapps.` For this example, Tomcat creates a `Tomcat\webapps\cc` directory. Each time you deploy a new copy of a `cc.war` file, delete the pre-existing `cc` directory structure created by the old `cc.war` file.

Installing a WebLogic Production Environment

For WebLogic, ClaimCenter is packaged into a WebLogic-specific EAR file. This topic explains the steps needed to configure WebLogic and deploy an EAR file.

Perform the steps described in the following sections:
- "Adding Servlet Definitions for WebLogic" on page 58
- "Generating the ClaimCenter EAR file for WebLogic" on page 58
- "Installing the EAR file on WebLogic" on page 59
- "Automatic Versus Manual Startup on WebLogic" on page 59

Adding Servlet Definitions for WebLogic

You might want to add definitions for additional servlets to the `web.xml` file.

To add servlet definitions

1. Copy the `ClaimCenter\modules\cc\deploy\WEB-INF\web.xml` file to `ClaimCenter\modules\configuration\deploy\WEB-INF\web.xml`.
2. Open `web.xml` in a text editor.
3. Add `servlet` definitions as needed. See the defined servlets for an example.
4. Add a `servlet-mapping` definition for each servlet that you add. See the defined servlet mappings for an example.
5. Save `web.xml`.

Generating the ClaimCenter EAR file for WebLogic

This procedure builds the ClaimCenter EAR file.

To generate the ClaimCenter EAR file

1. Open a command window.
2. Navigate to the ClaimCenter `bin` directory.
3. Run the following command:
   ```
   gwcc build-weblogic-ear
   ```
   This command generates the ClaimCenter EAR file in the ClaimCenter `dist\ear` directory.

Installing the EAR file on WebLogic

This procedure installs the generated ClaimCenter EAR file onto the WebLogic server.

To install the ClaimCenter EAR file

1. If WebLogic is not already running, start it now. If WebLogic is running, restart it.
2. After the server is running, point your browser to `http://localhost:7001/console.`

   Note: Port 7001 is the default port for WebLogic. If you configured WebLogic with a different port number, then change the port number in the address to the correct one.
3. Log in with your user name and password. The default WebLogic user name and password is `weblogic`.
4. On the left side of the user interface, under **Domain Structure**, click **Deployments.**
5. On the next screen, above **Domain Structure**, click **Lock & Edit**.
6. Within the main panel, under **Deployments**, click **Install.**
7. Navigate to the EAR file you generated and click **Next.**
8. Select **Install this deployment as an application** and click **Next**.
9. In the **Source accessibility** section of the next screen, select **I will make the deployment accessible from the following location.**
10. For the location, enter the path to the ClaimCenter `webapps` directory and click **Next.**
11. Click **Yes, take me to the deployment's configuration screen** and click **Finish.**
12. Review your configuration, and click **Activate Changes** located on the left side of the screen, above **Domain Structure.**

Automatic Versus Manual Startup on WebLogic

If you have configured WebLogic to automatically start (or restart) applications, WebLogic restarts ClaimCenter automatically on startup or on restart if the WebLogic server ever goes down. However, if you have not configured WebLogic to automatically start (or restart) applications, then start ClaimCenter manually. See "Starting ClaimCenter on WebLogic" on page 61.

Using HTTP Authentication on WebLogic

To authenticate using HTTP authentication on WebLogic, add the following inside the `security-configuration` element of the WebLogic `config.xml`:

```
<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
```

Installing a WebSphere Production Environment

To deploy a production instance of ClaimCenter on WebSphere, first add a welcome file list and add any necessary servlets to `web.xml`. Then, generate the WebSphere-specific ClaimCenter EAR file and install the EAR file on WebSphere.

Notes:

- Guidewire only supports externally managed data sources using JDBC drivers shipped with ClaimCenter and not the JDBC drivers that might be installed by default with the application server.
- Use the correct port number when you connect to the server in Studio. WebSphere's default port is 9080.
- Have the correct version of WebSphere. See "Configuring the Application Server" on page 13.

To add a welcome-file-list

This section explains how to add a `wecome-file-list` to the ClaimCenter `web.xml` file. If you do not perform this step, WebSphere cannot find your `index.html` file. Pointing your browser to: `http://localhost:9080/cc` does not work. Point to: `http://localhost:9080/cc/Start.do`.

1. Copy the `ClaimCenter\modules\cc\deploy\WEB-INF\web.xml` file to `ClaimCenter\modules\configuration\deploy\WEB-INF\web.xml`.
2. Open `web.xml` in a text editor.
3. At the bottom of the file, just before the `</web-app>` tag, add the following:

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

4. Add `servlet` definitions as needed. See the defined servlets for an example.
5. Add a `servlet-mapping` definition for each servlet that you add. See the defined servlet mappings for an example.
6. Save your changes and close `web.xml`.

To generate the ClaimCenter EAR File for WebSphere

1. Open a command window.
2. Navigate to the ClaimCenter `bin` directory.
3. Run the following command:

```
gwcc build-websphere-ear
```

   The command builds the EAR file and places it in `ClaimCenter\dist\ear`. The `build-websphere-ear` command packages the version of `web.xml` from the `ClaimCenter\modules\configuration\deploy\WEB-INF` directory.

To install the ClaimCenter EAR file on WebSphere

1. If WebSphere is not already running, start the application server.
2. Open the WebSphere Administrative Console.
3. From the left menu, expand **Applications**.
4. Click **New Application**.
5. Select **New Enterprise Application.**
6. Click **Browse** and select the ClaimCenter EAR file in the `ClaimCenter\dist\ear` directory.
7. Click **Next.**
8. Select **Fast Path.**
9. Click **Next.**
10. Accept the default installation options.

11. Click **Next**.
12. On the **Map modules to servers** page, verify that your targeted server or cluster is selected.
13. Click **Next**.
14. On the **Map virtual hosts for Web modules** page, verify that **default_host** is selected for the **Virtual host**.
15. Click **Next**.
16. On the **Summary** page, review your selections for accuracy. Click **Previous** to change any settings.
17. Click **Finish**.
18. Click **Save** to apply the changes to the master configuration.
19. Restart WebSphere.

After you have finished the procedure to install the ClaimCenter EAR file on WebSphere, proceed to “Starting ClaimCenter on WebSphere” on page 62.

Starting ClaimCenter on the Application Server

Select the link for your application server type to learn how to start ClaimCenter:
- “Starting ClaimCenter on JBoss” on page 61
- “Starting ClaimCenter on Tomcat on Windows” on page 61
- “Starting ClaimCenter on WebLogic” on page 61
- “Starting ClaimCenter on WebSphere” on page 62

For information on determining and setting the server mode, see:
- “Determining Server Mode” on page 62
- “Changing Server Mode” on page 63

Starting ClaimCenter on JBoss

ClaimCenter starts when you start JBoss. Use the `run` command in the JBoss `bin` directory to start JBoss. Entering the run command without any parameters launches JBoss using the default server configuration.

Starting ClaimCenter on Tomcat on Windows

ClaimCenter starts when you start Tomcat. To start the Tomcat server on Windows, run the following script:

```
Tomcat\bin\startup.bat
```

By default, Tomcat starts up in a new window. If it encounters errors, the new window might automatically close too quickly for you to read any error messages. In this case, you can run Tomcat in the same command window by editing the startup script. Locate the line in the script that runs Tomcat:

```
startup.bat: call "%EXECUTABLE%" start %CMD_LINE_ARGS%
```

Change the `start` option in the command to `run`. Save the script, and then run it again. Tomcat then runs in the same command window, which you can use to view any error messages.

Starting ClaimCenter on WebLogic

To start ClaimCenter on WebLogic

1. Start the WebLogic Admin Server if it is not already running.
2. Open the WebLogic Administration Console.

3. Under **Domain Structure**, click **Deployments**.
4. Select the checkbox for ClaimCenter.
5. Click **Start** → **Servicing all requests**. The WebLogic Administration Console informs you of the deployments that you selected to be started.
6. Click **Yes**.
7. Navigate to and select the ClaimCenter application. WebLogic identifies ClaimCenter by the name that you gave to the deployed EAR file.
8. Click **Deploy Application**.

   This step launches the ClaimCenter server. The application start process could take a few minutes to complete.

Starting ClaimCenter on WebSphere

To start ClaimCenter on WebSphere

1. Open the WebSphere Administrative Console.
1. Click **Applications** → **Application Types** → **WebSphere enterprise applications**.
2. Select the checkbox next to the ClaimCenter application, abbreviated by default as cc.

   If ClaimCenter is already running, and you want to restart ClaimCenter, restart WebSphere.

   If ClaimCenter is not running, click **Start**. ClaimCenter might take a few minutes to start.

IMPORTANT Guidewire does not support stopping and restarting the ClaimCenter server with the WebSphere Application Server (WAS) tools alone. Instead, to stop the ClaimCenter server, shutdown the WebSphere server itself.

Determining Server Mode

With all application server types except QuickStart, ClaimCenter can run in either development, test or production mode. Only development mode is available with QuickStart.

Guidewire provides these modes as a safety precaution so that development tools are not used on a production server. Some system functions are useful for development, but are not appropriate, or even dangerous, if used in a production environment. In development and test mode, both the **System Tools** and **Internal Tools** tabs are available. In production mode, *only* the **Server Tools** tab is available. See “Using Server and Internal Tools” on page 151 in the *System Administration Guide*.

An example is the `ITestingClock` plugin that provides functionality for setting the current time and is critical for testing time-sensitive processes. You can also use this plugin to modify the current time in a running server for demonstrations. However, use of this plugin in a production environment could have disastrous results. Therefore, you can only use this plugin when the server is in development or test mode. Test mode is identical to production mode except that in test mode you can adjust the testing system clock. See “Testing Clock Plugin (Only For Non-Production Servers)” on page 336 in the *Integration Guide*.

The ClaimCenter server can also be put into `MULTIUSER`, `DAEMONS`, and `MAINTENANCE` run levels. See “Using the Maintenance Run Level” on page 64 in the *System Administration Guide*. These run levels are independent of the mode. The combination of mode and run level determines the availability of functionality, such as the user interface and web services. For details, see “Understanding Server Run Levels and Modes” on page 62 in the *System Administration Guide*.

By default, ClaimCenter starts in production mode on all supported application servers other than the QuickStart server. ClaimCenter on the QuickStart server always runs in development mode. You cannot run ClaimCenter on the QuickStart server in production or test mode.

Verify the ClaimCenter mode by reading the console log as you start ClaimCenter or by checking the browser title bar.

Note: Whenever the server starts in development mode, ClaimCenter logs a warning.

Changing Server Mode

You can change the server mode while using any application server type except QuickStart. The QuickStart server always runs ClaimCenter in development mode.

Control the mode through the system parameter:

```
-Dgw.server.mode=(dev|prod|test)
```

To change the mode, restart the server and set `dev`, `test` or `prod` to the `-Dgw.server.mode` parameter:

```
-Dgw.server.mode=dev
```

or

```
-Dgw.server.mode=test
```

or

```
-Dgw.server.mode=prod
```

ClaimCenter ignores this parameter on the QuickStart server.

Connecting to ClaimCenter with a Web Client

Users connect to ClaimCenter through a web browser. The URL for ClaimCenter or ContactCenter includes the server name, port, and application name. For example:

```
http://appserver1:8080/cc/ClaimCenter.do
```

The following list shows default port numbers used by the application servers supported by ClaimCenter. You can configure the application server to listen on a different port than the default.

QuickStart	8080 (ClaimCenter)
	8280 (ContactCenter)
JBoss	8080
Tomcat	8080
WebLogic	7001
WebSphere	9080

Supply users with a user name and password along with the URL for your installation.

See “Client Information” on page 27 for a list of required software and hardware for client computers accessing ClaimCenter.

Changing the Superuser Password

ClaimCenter automatically creates an unrestricted superuser named `su` with full permissions. The default password for this superuser is `gw`. You create other users with the superuser account. Guidewire *strongly* recommends that when you start ClaimCenter for the first time, log in to ClaimCenter as user `su` and change this password.

To change the superuser password

1. Open a browser window.
2. Set the URL to the following:

   `http://server:port/cc/ClaimCenter.do`

   For example, if connecting on your local computer, use:

   `http://localhost:8080/cc/ClaimCenter.do`
3. Log into ClaimCenter as user `su` with password `gw`.
4. Click the **Preferences** link on the **Desktop** and change the password for user `su`.

You can change which user is the superuser. See “Changing the Unrestricted User” on page 34 in the *System Administration Guide*.

At this point, you have a running ClaimCenter server. However, there is no data in the installation and you have none of the tools available to manage the ClaimCenter server process. Before you use ClaimCenter, follow the processes defined in “Generating Java and SOAP API Libraries” on page 64, and “Connecting to ClaimCenter with a Web Client” on page 63.

Generating Java and SOAP API Libraries

ClaimCenter provides Java and SOAP APIs that you can use to integrate your own applications with ClaimCenter. These APIs are sometimes collectively referred to as the toolkit.

Generate these APIs when you first install ClaimCenter. Because the toolkit contains the ClaimCenter APIs, regenerate the toolkit after every data model change.

To generate the Java and SOAP APIs

1. From a command line, navigate to `ClaimCenter\bin`.
2. Execute the following command: `gwcc regen-java-api`.

   This command generates the `java-api` directory within the top-level ClaimCenter directory.
3. Execute the following command: `gwcc regen-soap-api`.

   This command generates the `soap-api` directory within the top-level ClaimCenter directory.

For more information, see “Integration Overview” on page 13 in the *Integration Guide*.

Enabling Integration between ClaimCenter and PolicyCenter

ClaimCenter includes a Gosu plugin that interfaces with a specialized PolicyCenter web service provided with PolicyCenter 3.0.0 or newer. The plugin enables ClaimCenter to retrieve policy information for certain policy types from PolicyCenter. Only Personal Auto and Workers’ Compensation policies are supported with this integration. PolicyCenter publishes the web service, `gw.webservice.pc.ccintegration.v2.CCPolicySearchIntegration` by default.

PolicyCenter includes a Gosu plugin that interfaces with a ClaimCenter web service provided with ClaimCenter 6.0. This plugin enables PolicyCenter to retrieve claim information from ClaimCenter.

This topic includes procedures to enable claim search integration from PolicyCenter and policy search integration from ClaimCenter.

To configure ClaimCenter to retrieve policy information from PolicyCenter

1. Start the PolicyCenter server.

2. Launch ClaimCenter Studio. From a command window, navigate to the `bin` directory within your ClaimCenter installation, and type `gwcc studio`.
3. In the Studio **Resources** pane, select **configuration** → **Plugins** → **gw** → **plugin** → **policy** → **search** → **IPolicySearchAdapter.**
4. Click **Remove** to remove the demonstration implementation, `gw.plugin.policy.impl.PolicySearchPluginDemoImpl`.
5. Click **Yes** when prompted to create a copy of `IPolicySearchAdapter` in the current module.
6. Click **Yes** when prompted to remove the plugin implementation.
7. Click Add.
8. Select **Gosu** for the plugin type.
9. Enter `gw.plugin.pcintegration.v2.PolicySearchPCPlugin` in the **Class** field.
10. Add user name and password parameters.
    a. Under **Parameters**, click **Add**.
    b. Enter the text `username` for the **Name**.
    c. Enter the superuser user name for the **Value**. By default, the superuser user name is `su`.
    d. Under **Parameters**, click **Add**.
    e. Enter the text `password` for the **Name**.
    f. Enter the superuser password for the **Value**. By default, the superuser password is `gw`.
11. Select **File** → **Save Changes**.
12. In the Studio **Resources** pane, select **configuration** → **Web Services** → **pcintegrationV2**.
13. Click **Edit**.
14. Click **Yes** when prompted to create a copy of `pcintegrationV2` in the current module.
15. Modify the **URL** to match the server and port on which PolicyCenter is running and remove the text `V2` from the WSDL URL.
16. Select **File** → **Save Changes**.
17. Click **Refresh** in the upper right corner of the web service window. You must refresh the web service, even if you have made no changes to the web service.
18. To test the integration, in ClaimCenter, start the New Claim Wizard and search for a Personal Auto or Workers' Compensation policy. The integration supports only Personal Auto and Workers' Compensation policies.

The ClaimCenter plugin implements the two methods on `IPolicySearchAdapter`: `searchForPolicies` and `retrievePolicy`. The PolicyCenter web service returns objects that look like ClaimCenter policy entities, so implementation of the plugin is relatively straightforward. The web service performs the conversion by translating to and from the soap objects used to communicate with the web service.

The conversion of objects received from PolicyCenter is configurable in the `pc-to-cc-data-mapping.xml` file. ClaimCenter Studio exposes this file under **configuration** → **Other Resources**. The only object ClaimCenter sends to PolicyCenter is `PCSearchCriteria`. This object is created and populated from the ClaimCenter search criteria using Gosu. Configuration of this translation must be done in Gosu.

Changes to the `pc-to-cc-data-mapping.xml` file are not picked up automatically by the plugin. After changing the file, either restart your ClaimCenter server, or run the following in the Gosu tester in ClaimCenter Studio while Studio is connected to the ClaimCenter server:

```
uses gw.plugin.pcintegration.impl.PolicySearchConverter
PolicySearchConverter.INSTANCE = new PolicySearchConverter()
```

To configure PolicyCenter to retrieve claim information from ClaimCenter

1. Start the ClaimCenter server.
2. Launch PolicyCenter Studio. From a command window, navigate to the `bin` directory within your PolicyCenter installation, and type `gwpc studio`.
3. In the Studio **Resources** pane, select **configuration** → **Web Services** → **CCClaimSearchAPI**.
4. Click **Edit**.
5. Click **Yes** when prompted to create a copy of `CCClaimSearchAPI` in the current module.
6. Modify the **URL** to match the server and port on which ClaimCenter is listening.
7. Select **File** → **Save Changes**.
8. Click **Refresh** in the upper right corner of the web service window. You must refresh the web service, even if you have made no changes to the web service.
9. In the Studio **Resources** pane, select **configuration** → **Plugins** → **gw** → **plugin** → **claimsearch** → **IClaimSearchPlugin**.
10. Click **Remove** to remove the demonstration implementation, `gw.plugin.claimsearch.impl.GWDemoClaimSearchPlugin`.
11. Click **Yes** when prompted to create a copy of `CCClaimSearchAPI` in the current module.
12. Click **Yes** when prompted to remove the plugin implementation.
13. Click Add.
14. Select **Gosu** for the plugin type.
15. Enter `gw.plugin.claimsearch.impl.GWClaimSearchPlugin` in the **Class** field.
16. Add user name and password parameters.
    a. Under **Parameters**, click **Add**.
    b. Enter the text `username` for the **Name**.
    c. Enter the superuser user name for the **Value**. By default, the superuser user name is `su`.
    d. Under **Parameters**, click **Add**.
    e. Enter the text `password` for the **Name**.
    f. Enter the superuser password for the **Value**. By default, the superuser password is `gw`.
17. Select **File** → **Save Changes**.
18. In the Studio **Resources** pane, select **configuration** → **Page Configuration (PCF)** → **exitpoints** → **ViewClaim**.
19. Select the `ClaimSystemURL` parameter.
20. Change the `urlParam` property to the following, substituting *`server`* and *`port`* with values for your ClaimCenter implementation:
    ```
    http://server:port/cc/ClaimSummaryLink.do
    ```
21. Select **File** → **Save Changes**.

After You Install ClaimCenter

This section discusses topics related to additional installation and configuration issues.

Enabling Archiving or Disabling Archive Work Queue

The default `config.xml` file has archiving disabled. This is set by the parameter:

```
<param name="ArchiveEnabled" value="false"/>
```

If you want to enable archiving, set the value of `ArchiveEnabled` to `true`. Then review the topics listed further in this section to learn how to configure archiving.

If you do not want to enable archiving, Guidewire recommends that you disable the archive work queue.

To disable the archive work queue

1. In a command window, navigate to the `ClaimCenter\bin` directory in the ClaimCenter installation.
2. Launch Guidewire Studio using the following command:
   ```
   gwcc studio
   ```
3. In Studio, open **Other Resources** → **work-queue.xml**.
4. Comment out the following block by adding `<!--` before the block and `-->` after it.
   ```
   <work-queue workQueueClass="com.guidewire.pl.domain.archiving.ArchiveWorkQueue"
    progressinterval="600000">
       <worker instances="1"/>
   </work-queue>
   ```
   If you have not previously edited `work-queue.xml`, click **Yes** when Studio prompts you to create a copy of `work-queue.xml` in the configuration module.
5. Select **File** → **Save Changes**.

See also

- "Archiving" on page 87 in the *Application Guide* – information on archiving claims, searching for archived claims, and restoring archived claims.
- "Archive Parameters" on page 39 in the *Configuration Guide* – discussion on the configuration parameters used in claims archiving.
- "Archiving Claims" on page 665 in the *Configuration Guide* – information on configuring claims archiving, selecting claims for archiving, and archiving and the object (domain) graph.
- "Archiving Integration" on page 285 in the *Integration Guide* – describes the archiving integration flow, storage and retrieval integration, and the `IArchiveSource` plugin interface.
- "Archive" on page 48 in the *Rules Guide* – information on base configuration archive rules and their use in detecting archive events and managing the claims archive and restore process.
- "Logging Successfully Archived Claims" on page 37 in the *System Administration Guide*.
- "Purging Unwanted Claims" on page 58 in the *System Administration Guide*.
- "Archive Info" on page 160 in the *System Administration Guide*.
- "Upgrading Archived Entities" on page 62 in the *Upgrade Guide*.

IMPORTANT To increase performance, most customers find increased hardware more cost effective than archiving unless their volume exceeds one million claims or more. Guidewire strongly recommends that you contact Customer Support before implementing archiving to help your company with this analysis.

Integrating ClaimCenter with ContactCenter

To integrate ClaimCenter with ContactCenter, see “Integrating ClaimCenter with ContactCenter” on page 11 in the *Contact Management Guide*.

Using Standard Reporting

If you are going to use Guidewire Standard Reporting, see the *ClaimCenter Reporting Guide*.

Running ClaimCenter in a Clustered Environment

Running ClaimCenter in a clustered environment requires an in depth understanding of ClaimCenter configuration files. The *ClaimCenter System Administration Guide* discusses the configuration environment. This guide also explains how to configure multiple ClaimCenter servers as a cluster.

Updating Your Installation

For information on upgrading your installation, see the *ClaimCenter Upgrade Guide*.

chapter 5

Commands Reference

This topic lists the commands that are used to operate ClaimCenter in the configuration environment.

Because of their simplicity and power they offer, command line tools are the preferred method of controlling server behavior, loading data, and generating tools in the Guidewire configuration environment. These commands can be configured using familiar files, are invoked using standard developer tools, and are compatible with a wide spectrum of development environments.

For a description of administrative commands provided with ClaimCenter, see "ClaimCenter Administrative Commands" on page 169 in the *System Administration Guide*.

This topic includes:

- "Tuning Command Line Tool Memory Settings" on page 69
- "QuickStart Command Tools" on page 70
- "Build Tools" on page 71

Tuning Command Line Tool Memory Settings

The `ClaimCenter\modules\pl\etc\memory.properties` file specifies memory settings for the QuickStart application server and other `gwcc` tools. To change one or more of these settings, copy the `ClaimCenter\modules\pl\etc\memory.properties` file to `ClaimCenter\modules\configuration\etc`. Then edit the `xms`, `xmx`, and `maxperm` values for the class that runs the tool. The class information is provided with the command lists in "QuickStart Command Tools" on page 70 and "Build Tools" on page 71.

For example, to change the memory settings for the QuickStart server, set the following properties in `ClaimCenter\modules\configuration\etc\memory.properties`:

- starting heap size: `com.guidewire.commons.jetty.GWServerJettyServerMain.xms`
- maximum heap size: `com.guidewire.commons.jetty.GWServerJettyServerMain.xmx`
- maximum permanent size: `com.guidewire.commons.jetty.GWServerJettyServerMain.maxperm`

QuickStart Command Tools

Use the following commands to control the QuickStart method of installing ClaimCenter:

Command	Action
`gwcc dev-debug-shmem`	Starts ClaimCenter in development mode using shared memory debugging.
`gwcc dev-debug-socket`	Starts ClaimCenter in development mode using socket debugging.
`gwcc dev-deploy`	Copies resources for the QuickStart application server.
`gwcc dev-dropdb`	Deletes the QuickStart database (usually contained in the `tmp` directory.) Class: `com.guidewire.testharness.db.DBResetTool`
`gwcc dev-start`	Starts the bundled QuickStart application server. Class: `com.guidewire.commons.jetty.GWServerJettyServerMain`
`gwcc dev-stop`	Stops the bundled QuickStart application server. Class: `com.guidewire.commons.jetty.GWServerJettyServerStopMain`
`gwcc dev-suspend-shmem`	Starts ClaimCenter in development mode using shared memory debugging. Starts suspended.
`gwcc dev-suspend-socket`	Start ClaimCenter in development mode using socket debugging. Starts suspended.

Build Tools

Use the following commands instead of any build scripts which accompany your IDE.

Command	Action
`gwcc -p`	Displays all `gwcc` command options.
`gwcc build-war`	Builds the generic WAR file for Tomcat. You can include the Boolean parameter `config.war.dictionary=true` to also generate the ClaimCenter *Data Dictionary* and *Security Dictionary* while building the WAR file. Use the following command: `gwcc build-war -Dconfig.war.dictionary=true` When `config.war.dictionary=true`, the command creates a `dictionary` folder within the WAR file. The `dictionary` folder contains `data` and `security` folders. These folders contain the *Data Dictionary* and *Security Dictionary* respectively. To view a dictionary, open `index.html` in the `data` or `security` folder.
`gwcc build-weblogic-ear`	Builds the EAR file for WebLogic.
`gwcc build-websphere-ear`	Builds the EAR file for WebSphere.
`gwcc copy-starter-resources`	Copies starter configuration resources to the configuration module, if applicable. Class: `com.guidewire.tools.config.CopyStarterConfigResourcesTool`
`gwcc regen-datamapping-split`	Builds the data mapping files with files split out by table and typelist. Class: `com.guidewire.tools.datamapping.DataMappingTool`
`gwcc regen-datamapping-together`	Builds the data mapping files with all tables and typelists concatenated. Class: `com.guidewire.tools.datamapping.DataMappingTool`
`gwcc regen-dictionary`	Generates the *ClaimCenter Data Dictionary* and *ClaimCenter Security Dictionary*. Generate the first time you unzip the application and each time you update the data model. Run the `gwcc regen-java-api` and `gwcc regen-soap-api` commands each time just prior to regenerating the security and data dictionaries. To view either, open a new browser window and enter: • `ClaimCenter/dictionary/data/index.html` for the *Data Dictionary* or • `ClaimCenter/dictionary/security/index.html` for the *Security Dictionary*. You can also generate these dictionaries while building a WAR file. See the description for `gwcc build-war` for instructions. Classes: `com.guidewire.tools.dictionary.data.DataDictionaryTool` `com.guidewire.tools.dictionary.security.SecurityDictionaryTool`
`gwcc regen-gosudoc`	Generates Gosu documentation similar to JavaDoc but using the ClaimCenter type system. This command produces documentation at `ClaimCenter/build/gosudoc/index.html`. See "Gosu Generated Documentation" on page 35 in the *Gosu Reference Guide*. Class: `com.guidewire.tools.gosudoc.GosuDocMain`
`gwcc regen-java-api`	Builds the Java API toolkit and examples to the `ClaimCenter/java-api` directory. See "Regenerating the Integration Libraries" on page 17 in the *Integration Guide*. Class: `com.guidewire.commons.entity.external.ExternalGenerator`
`gwcc regen-pcfmapping`	Builds the PCF mappings. Class: `com.guidewire.tools.pcfmapping.PCFMappingWriterMain`
`gwcc regen-rulereport`	Generates an XML report describing the existing business rules. See "Generating a Rule Repository Report" on page 91 in the *Rules Guide*.
`gwcc regen-soap-api`	Builds the SOAP API toolkit and examples to the `ClaimCenter/soap-api` directory. See "Regenerating the Integration Libraries" on page 17 in the *Integration Guide*. Classes: `com.guidewire.tools.wsdl.WSDLGenerator` `com.guidewire.util.webservices.axis.WSDLToJavaGenerator`

Command	Action
`gwcc regen-xsd`	Builds the XSD files for data import. See “Creating an XML File for Import” on page 121 in the *System Administration Guide* and “Importing Administrative Data” on page 83 in the *Integration Guide*.
`gwcc studio`	Runs Guidewire Studio. Class: `com.guidewire.studio.main.Main`
`gwcc verify-checksum`	Verifies module checksums. Class: `com.guidewire.tools.checksum.ModulesChecksumTool`
`gwcc verify-types`	Checks PCF files for errors. See“Validating Studio Resources” on page 121 in the *Configuration Guide*.
`gwcc version`	Displays the product version.