

Guidewire PolicyCenter®

PolicyCenter Installation Guide

RELEASE 8.0.4

Copyright © 2001-2015 Guidewire Software, Inc. All rights reserved.

Guidewire, Guidewire Software, Guidewire ClaimCenter, Guidewire PolicyCenter, Guidewire BillingCenter, Guidewire Reinsurance Management, Guidewire ContactManager, Guidewire Vendor Data Management, Guidewire Client Data Management, Guidewire Rating Management, Guidewire InsuranceSuite, Guidewire ContactCenter, Guidewire Studio, Guidewire Product Designer, Guidewire Live, Guidewire DataHub, Guidewire InfoCenter, Guidewire Standard Reporting, Guidewire ExampleCenter, Guidewire Account Manager Portal, Guidewire Claim Portal, Guidewire Policyholder Portal, Gosu, Deliver Insurance Your Way, and the Guidewire logo are trademarks, service marks, or registered trademarks of Guidewire Software, Inc. in the United States and/or other countries.

All other trademarks are the property of their respective owners.

This material is confidential and proprietary to Guidewire and subject to the confidentiality terms in the applicable license agreement and/or separate nondisclosure agreement.

Guidewire products are protected by one or more United States patents.

Product Name: Guidewire PolicyCenter

Product Release: 8.0.4

Document Name: PolicyCenter Installation Guide

Document Revision: 25-May-2015

Contents

About PolicyCenter Documentation	7
Conventions in This Document	8
Support	8
1 Introduction to Installation	9
Selecting an Installation Scenario	10
List of Installation Options	11
Viewing the Development and Production Environments	13
2 Preparing a PolicyCenter Environment	15
Installation Environments Overview	15
Production Environments	16
Development Environments	16
Creating Accounts to Run PolicyCenter	16
Configuring the Application Server	17
General Guidelines for the Application Server	17
Considerations for a Clustered Application Server Environment	18
Configuring the JVM for Environments Without Graphics	19
JVM Heap Size Considerations	19
Load Balancers	21
Optional Components	21
Configuring JBoss	21
Configuring Tomcat	22
Configuring WebLogic	24
Configuring WebSphere	25
Configuring the Database	27
Database Permissions	28
Configuring Linguistic Search Collation	28
Configuring Compression	28
Configuring Oracle for PolicyCenter	32
Configuring SQL Server for PolicyCenter	38
Development Workstation Information	42
Client Information	43
Enabling DOM Storage	43
Installing Java	43
Installing the Dynamic Code Evolution Virtual Machine	44
Installing Ant	45
Setting Environment Variables	45
Documenting Your Environment	46
3 Installing a PolicyCenter Development Environment	47
Using Multiple PolicyCenter Development Instances	48
Installing the QuickStart Development Environment	48
Advantages to Using the QuickStart Software	48
QuickStart Development Environment Prerequisites	49
Installing with QuickStart	49
QuickStart Commands	50
QuickStart Application Server	50
Troubleshooting QuickStart	51

Installing a Tomcat Development Environment	51
Tomcat Development Environment Prerequisites	51
Building a PolicyCenter Development Environment with Tomcat	51
Using the QuickStart Database	53
Setting the Database Configuration and Mode	53
Limitations to the QuickStart Database	54
Additional References	54
Using SQL Server or Oracle in a Development Environment	54
Configuring Archiving for Development Testing	54
Enabling Reinsurance Management or Disabling Work Queue	55
Installing Sample Data	55
Using Gosu to Configure Sample Data	56
4 Installing a PolicyCenter Production Environment	59
Unpacking the Configuration Files	59
Key PolicyCenter gwpc Commands	61
Configuring a Database Connection	66
The <database> Element	67
Mapping Logical Tablespaces to Physical Tablespaces	68
Configuring Options for Individual Tables	69
Defining Table Groups	70
Defining the JDBC URL	71
Specifying a Database Password	71
Enabling SQL Server JDBC Logging	72
Configuring PolicyCenter to Use a JNDI Data Source	73
Creating a JNDI Data Source on JBoss	74
Creating a JNDI Data Source on Tomcat	76
Creating a JNDI Data Source on WebLogic	77
Creating a JNDI Data Source on WebSphere	79
Deploying PolicyCenter to the Application Server	84
Installing a JBoss Production Environment	84
Installing a Tomcat Production Environment	84
Installing a WebLogic Production Environment	85
Installing a WebSphere Production Environment	87
5 Additional PolicyCenter Setup Tasks	89
Free-text Search Setup	89
Free-text Search Setup Overview	90
Setting Up Free-text Search for Embedded Operation	92
Setting Up Free-text Search for JBoss	93
Setting Up Free-text Search for Tomcat	95
Setting Up Free-text Search for WebLogic	96
Setting Up Free-text Search for WebSphere	97
Setting Up the Free-text Batch Load Command	100
Changing the Superuser Password	102
Generating Java and SOAP API Libraries	102
Enabling Integration between ClaimCenter and PolicyCenter	102
Enabling Large Loss Notification Integration	106
Enabling Integration between BillingCenter and PolicyCenter	107
Verifying the BillingCenter Integration	110
Enabling Archiving or Disabling Archiving Work Queues	111
Installing Rating Management	112
Installing Reinsurance Management or Disabling Work Queue	112
Upgrading Product Model of Old LOB Extension Packs	113
Configuring Single Sign-on Authentication	113

Starting PolicyCenter on the Application Server	116
Starting PolicyCenter on JBoss	116
Starting PolicyCenter on Tomcat on Windows	116
Starting PolicyCenter on WebLogic	117
Starting PolicyCenter on WebSphere	117
Connecting to PolicyCenter with a Web Client	117
Configuring Windows Accessibility for Firefox	118
Additional Installation Information	118
Integrating PolicyCenter with ContactManager	118
Running PolicyCenter in a Clustered Environment	118
6 Commands Reference	119
Tuning Command Line Tool Memory Settings	119
QuickStart Command Tools	120
Build Tools	121

About PolicyCenter Documentation

The following table lists the documents in PolicyCenter documentation.

Document	Purpose
<i>InsuranceSuite Guide</i>	If you are new to Guidewire InsuranceSuite applications, read the <i>InsuranceSuite Guide</i> for information on the architecture of Guidewire InsuranceSuite and application integrations. The intended readers are everyone who works with Guidewire applications.
<i>Application Guide</i>	If you are new to PolicyCenter or want to understand a feature, read the <i>Application Guide</i> . This guide describes features from a business perspective and provides links to other books as needed. The intended readers are everyone who works with PolicyCenter.
<i>Upgrade Guide</i>	Describes how to upgrade PolicyCenter from a previous major version. The intended readers are system administrators and implementation engineers who must merge base application changes into existing PolicyCenter application extensions and integrations.
<i>New and Changed Guide</i>	Describes new features and changes from prior PolicyCenter versions. Intended readers are business users and system administrators who want an overview of new features and changes to features. Consult the "Release Notes Archive" part of this document for changes in prior maintenance releases.
<i>Installation Guide</i>	Describes how to install PolicyCenter. The intended readers are everyone who installs the application for development or for production.
<i>System Administration Guide</i>	Describes how to manage a PolicyCenter system. The intended readers are system administrators responsible for managing security, backups, logging, importing user data, or application monitoring.
<i>Configuration Guide</i>	The primary reference for configuring initial implementation, data model extensions, and user interface (PCF) files. The intended readers are all IT staff and configuration engineers.
<i>Globalization Guide</i>	Describes how to configure PolicyCenter for a global environment. Covers globalization topics such as global regions, languages, date and number formats, names, currencies, addresses, and phone numbers. The intended readers are configuration engineers who localize PolicyCenter.
<i>Rules Guide</i>	Describes business rule methodology and the rule sets in PolicyCenter Studio. The intended readers are business analysts who define business processes, as well as programmers who write business rules in Gosu.
<i>Contact Management Guide</i>	Describes how to configure Guidewire InsuranceSuite applications to integrate with ContactManager and how to manage client and vendor contacts in a single system of record. The intended readers are PolicyCenter implementation engineers and ContactManager administrators.
<i>Best Practices Guide</i>	A reference of recommended design patterns for data model extensions, user interface, business rules, and Gosu programming. The intended readers are configuration engineers.
<i>Integration Guide</i>	Describes the integration architecture, concepts, and procedures for integrating PolicyCenter with external systems and extending application behavior with custom programming code. The intended readers are system architects and the integration programmers who write web services code or plugin code in Gosu or Java.
<i>Gosu Reference Guide</i>	Describes the Gosu programming language. The intended readers are anyone who uses the Gosu language, including for rules and PCF configuration.
<i>Glossary</i>	Defines industry terminology and technical terms in Guidewire documentation. The intended readers are everyone who works with Guidewire applications.

Document	Purpose
<i>Product Model Guide</i>	Describes the PolicyCenter product model. The intended readers are business analysts and implementation engineers who use PolicyCenter or Product Designer. To customize the product model, see the <i>Product Designer Guide</i> .
<i>Product Designer Guide</i>	Describes how to use Product Designer to configure lines of business. The intended readers are business analysts and implementation engineers who customize the product model and design new lines of business.

Conventions in This Document

Text style	Meaning	Examples
<i>italic</i>	Emphasis, special terminology, or a book title.	A <i>destination</i> 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. In code blocks, bold formatting highlights relevant sections to notice or to configure.	Get the field from the Address object.
<i>monospaced italic</i>	Parameter names or other variable placeholder text within URLs or other code snippets.	Use <code>getName(<i>first</i>, <i>last</i>)</code> . <code>http://SERVERNAME/a.html</code> .

Support

For assistance, visit the Guidewire Resource Portal – <http://guidewire.custhelp.com>

Introduction to Installation

This guide describes how to install PolicyCenter. It describes installation options that are available and provides guidance on selecting which option to choose.

The *PolicyCenter Installation Guide* includes:

- An overview that discusses the different ways to install PolicyCenter that starts with this topic.
- “Preparing a PolicyCenter Environment” on page 15 provides steps to prepare a development or production environment for PolicyCenter.
- “Installing a PolicyCenter Development Environment” on page 47 explains how to install a PolicyCenter development environment using the QuickStart server and database or Tomcat.
- “Installing a PolicyCenter Production Environment” on page 59 explains how to deploy PolicyCenter to an application server and database server production environment.
- “Additional PolicyCenter Setup Tasks” on page 89 explains optional installation tasks you may want to perform after you complete the installation and deployment of a PolicyCenter development or production environment.
- “Commands Reference” on page 119 lists and describes the QuickStart and build commands. Other PolicyCenter command utilities are described in “PolicyCenter Administrative Tools” on page 173 in the *System Administration Guide*.

This topic includes:

- “Selecting an Installation Scenario” on page 10
- “Viewing the Development and Production Environments” on page 13

Selecting an Installation Scenario

Choose an installation scenario based on the role of the person who uses the PolicyCenter 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 15 “Installing a PolicyCenter Development Environment” on page 47
Application Developer	Changes the behavior of the application including the user interface, rules, and application logic.	“Installation Environments Overview” on page 15 “Installing a PolicyCenter Development Environment” on page 47
Integration Developer	Develops software to connect PolicyCenter to external systems.	“Installation Environments Overview” on page 15 “Installing a PolicyCenter Development Environment” on page 47
Conversion Developer	Performs analysis and mapping of legacy data structures to Guidewire application data model.	“Installation Environments Overview” on page 15 “Installing a PolicyCenter Development Environment” on page 47
Build master deploying to testing and production	Deploys finished application to test and production environments.	“Installation Environments Overview” on page 15 “Installing a PolicyCenter Production Environment” on page 59.

List of Installation Options

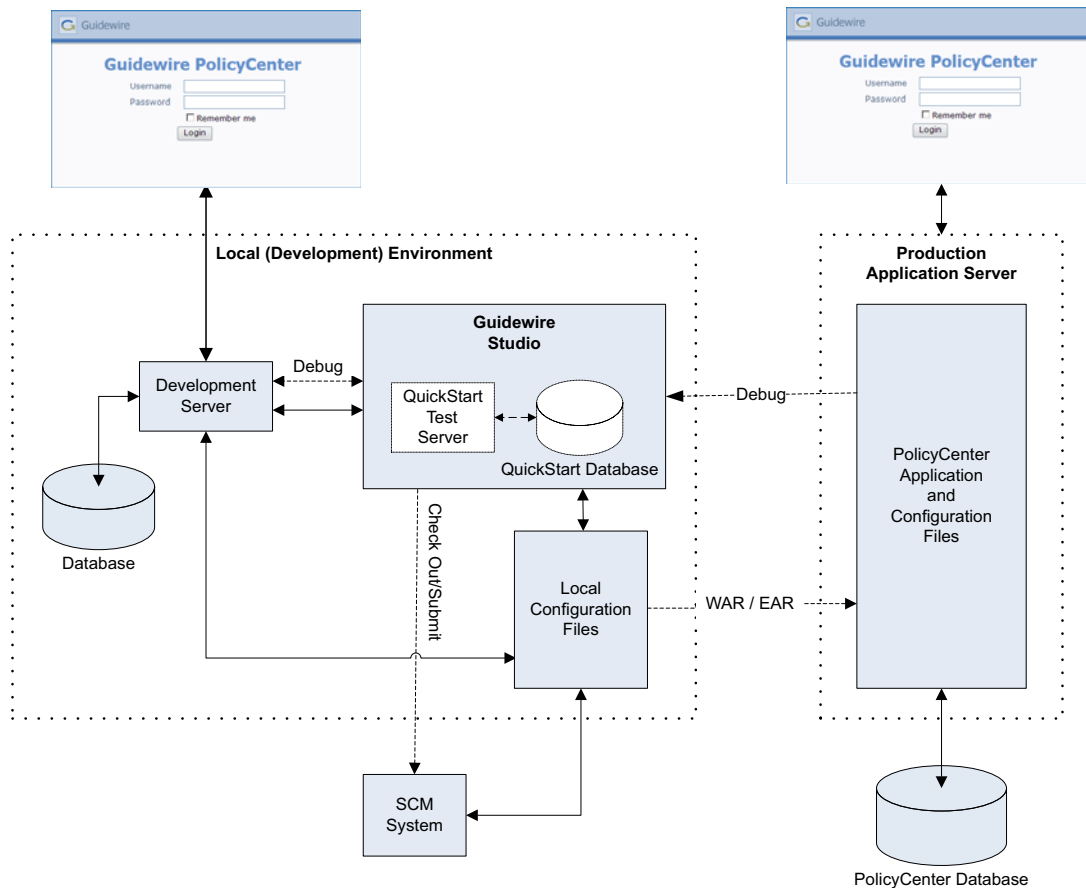
If you are still questioning your installation options based on your role, this section can provide additional clarification.

Install element	Bundled or Optional	Good to know	Links for more Information
QuickStart Application Server	Bundled	<p>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.</p> <p>The QuickStart application server can be used for demonstration or development environments. Guidewire does not support QuickStart for a production environment.</p> <p>PolicyCenter can be quickly started in development (dev) mode using the steps in "Installing the QuickStart Development Environment" on page 48. PolicyCenter can not run in production (prod) mode on QuickStart. For more information on server modes, see "Server Modes and Run Levels" on page 58 in the <i>System Administration Guide</i>.</p>	<p>"Advantages to Using the QuickStart Software" on page 48</p> <p>"QuickStart Application Server" on page 50</p>
JBoss Application Server	Optional	Suitable for production environments.	<p>"Configuring the Application Server" on page 17</p> <p>"Installing a JBoss Production Environment" on page 84</p>
Tomcat Application Server	Optional	<p>Suitable for production environments.</p> <p>You can use Tomcat instead of the QuickStart server for development work although it requires additional configuration.</p>	<p>"Configuring the Application Server" on page 17</p> <p>"Installing a Tomcat Development Environment" on page 51</p> <p>"Installing a Tomcat Production Environment" on page 84</p>
WebSphere Application Server	Optional	Suitable for production environments.	<p>"Configuring the Application Server" on page 17</p> <p>"Installing a WebSphere Production Environment" on page 87</p>
WebLogic Application Server	Optional	Suitable for production environments.	<p>"Configuring the Application Server" on page 17</p> <p>"Installing a WebLogic Production Environment" on page 85</p>

Install element	Bundled or Optional	Good to know	Links for more Information
QuickStart Database	Bundled	<p>You can immediately use the QuickStart database in file mode. PolicyCenter creates and stores the database files within the tmp directory of the local drive. You can customize this location.</p> <p>You can use the QuickStart database for demonstration or development environments. Guidewire does not support QuickStart for a production environment.</p> <p>Guidewire does not support upgrades to the QuickStart database. Configuring your application sometimes requires extending the data model, which might require dropping the database.</p> <p>You can not have more than one connection at a time.</p>	<p>"Advantages to Using the QuickStart Software" on page 48</p> <p>"Using the QuickStart Database" on page 53</p>
Oracle Database	Optional	You can use Oracle for development and production.	<p>"Configuring the Database" on page 27</p> <p>"Configuring Oracle for PolicyCenter" on page 32</p> <p>"Configuring a Database Connection" on page 66</p>
SQL Server Database	Optional	You can use SQL Server for development and production.	<p>"Configuring the Database" on page 27</p> <p>"Configuring SQL Server for PolicyCenter" on page 38</p> <p>"Configuring a Database Connection" on page 66</p>

Viewing the Development and Production Environments

The following diagram illustrates how the PolicyCenter 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 PolicyCenter 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 PolicyCenter 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.

Preparing a PolicyCenter Environment

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

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>.

This topic includes:

- “Installation Environments Overview” on page 15
- “Creating Accounts to Run PolicyCenter” on page 16
- “Configuring the Application Server” on page 17
- “Configuring the Database” on page 27
- “Development Workstation Information” on page 42
- “Client Information” on page 43
- “Installing Java” on page 43
- “Installing Ant” on page 45
- “Setting Environment Variables” on page 45
- “Documenting Your Environment” on page 46

Installation Environments Overview

PolicyCenter 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 PolicyCenter. 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, Development environments must be on Windows. However, you can build PolicyCenter on a non-Development Unix system prior to deploying PolicyCenter.

See also

- “Development Workstation Information” on page 42
- “Build Scripts Supported on Unix” on page 127

Production Environments

Guidewire strongly recommends that you allocate dedicated hardware for the application server and database server tiers. Reserve hardware solely for PolicyCenter. 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 PolicyCenter on a Windows system prior to deploying PolicyCenter.

PolicyCenter 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 PolicyCenter Environment” on page 15. Then proceed to “Installing a PolicyCenter Production Environment” on page 59.

Development Environments

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

For development, all builds must use the Oracle JDK. See “Development Workstation Information” on page 42.

For all development environments, review the following topics:

- “Development Workstation Information” on page 42
- “Client Information” on page 43
- “Creating Accounts to Run PolicyCenter” on page 16 (Windows information only)
- “Installing Java” on page 43
- “Installing Ant” on page 45
- “Setting Environment Variables” on page 45
- “Documenting Your Environment” on page 46

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

- “Configuring the Application Server” on page 17
- “Configuring Tomcat” on page 22

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

- “Configuring the Database” on page 27

After reviewing the relevant development environment information, proceed to “Installing a PolicyCenter Development Environment” on page 47.

Creating Accounts to Run PolicyCenter

It is important that the software processes that support your PolicyCenter application run with the appropriate permissions. How you set up these accounts depends on whether the application server environment is 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 to ensure that PolicyCenter is run with the correct rights.

If you run Tomcat on Microsoft Windows, install the PolicyCenter server as a Windows service. See “Installing Tomcat as a Windows Service” on page 23 for more information.

For a UNIX-based operating system, the PolicyCenter-related processes must run in non-privileged (user) mode. A process in non-privileged mode can access only its own memory. To ensure the PolicyCenter 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 PolicyCenter 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 PolicyCenter 8.0.4. 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 50.

This topic includes the following:

- “General Guidelines for the Application Server” on page 17
- “Considerations for a Clustered Application Server Environment” on page 18
- “Configuring the JVM for Environments Without Graphics” on page 19
- “JVM Heap Size Considerations” on page 19
- “Load Balancers” on page 21
- “Optional Components” on page 21
- “Configuring JBoss” on page 21
- “Configuring Tomcat” on page 22
- “Configuring WebLogic” on page 24
- “Configuring WebSphere” on page 25

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.

PolicyCenter synchronizes with the database clock. The application server and database server must be in 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.

See also

- “Clustering Application Servers” on page 77 in the *System Administration Guide*
- “Clustering Parameters” on page 44 in the *Configuration Guide*

Clustering Parameters

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

- `ClusteringEnabled`
- `ClusterMemberPurgeDaysOld`
- `ClusterMemberRecordUpdateIntervalSecs`
- `ClusterMulticastAddress`
- `ClusterMulticastPort`
- `ClusterMulticastTTL`
- `ClusterProtocolStack`
- `ClusterProtocolStackOption1`
- `ClusterProtocolStackOption2`
- `ClusterStatisticsMonitorIntervalMins`
- `ConfigVerificationEnabled`
- `JGroupsClusterChannel`
- `JGroupsWatchdogHeartbeatIntervalSecs`
- `JGroupsWatchdogMissedHeartbeatsBeforeReset`
- `PDFMergeHandlerLicenseKey`

Using ENCRYPT JGroups Protocol

If a cluster is set up in different subnets and there is a risk of traffic sniffing between these subnets, Guidewire recommends using the ENCRYPT JGroups protocol to encrypt messages. This protocol may be configured using the `ClusterProtocolStack` parameter by including it in the JGroups protocol stack.

Refer to <http://jgroups.org/manual/html/protolist.html#d0e6465>.

Using TCP Instead of UDP

By default, PolicyCenter uses UDP (User Datagram Protocol) for communication between servers in a clustered environment. For a cluster with fewer than ten nodes, you can use TCP (Transmission Control Protocol) if you prefer. The TCP cluster protocol stack is different from the UDP version because some protocols do not work well without native broadcast support. To use TCP, set the `ClusterProtocolStack` parameter in `config.xml` to the following:

```
<param name="ClusterProtocolStack"
  value="TCP(bind_port=${jgroups.bind_port};bind_addr=${jgroups.bind_addr}):
gw.JDBC_PING(timeout=5000;num_initial_members=4;num_ping_requests=3;updateInterval=30000):
MERGE2(max_interval=30000;min_interval=10000):
FD(timeout=5000;max_tries=5):
VERIFY_SUSPECT(timeout=3000;num_msgs=3):
pbcast.NAKACK(retransmit_timeout=600,1200,2400,4800;discard_delivered_msgs=true):
```

```
UNICAST():
pbcast.STABLE(desired_avg_gossip=5000;max_bytes=4M):
pbcast.GMS(join_timeout=6000;merge_timeout=10000;print_local_addr=true)
```

Pass two JVM parameters when starting the server: `-Djgroups.bind_addr` and `-Djgroups.bind_port`. The values you pass replace the corresponding placeholders in `config.xml`. Set `jgroups.bind_addr` to the IP address of the batch server. Set `-Djgroups.bind_port` to the port you want to use for cluster communication.

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 PolicyCenter. 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, PolicyCenter uses JGroups for some multicast communication between servers in the cluster. In a clustered RedHat environment, the default network setup does not enable proper multicast communication in 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 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. Setting `java.awt.headless` to `true` prevents the JVM from attempting to access a 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 23
- “Increasing Heap Size for WebLogic” on page 24
- “Increasing Heap Size for WebSphere” on page 25

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 that 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.

The following tables listing limits on heap size serve only as a starting point. They are useful if you are installing PolicyCenter 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
Windows	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	1.5 GB	Very large

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 PolicyCenter. 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
initial heap size	Xms	1 GB	2 GB
maximum heap size	Xmx	1 GB	2 GB
maximum permanent generation size	MaxPermSize	128 MB	256 MB

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. PolicyCenter supports both sessionless and sessioned SOAP calls, the latter of which require IP affinity.

Optional Components

If you plan to have PolicyCenter 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 PolicyCenter. 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 topic provides notes on installing JBoss to run PolicyCenter.

Removing JBoss JARs

Remove `modules/system/layers/base/org/codehaus/woodstox/main/stax2-api-3.1.1-redhat-3.jar` from your JBoss installation, if present. The `stax2-api-3.1.1-redhat-3.jar` conflicts with PolicyCenter in the JBoss class loader. Not all JBoss versions include this JAR file.

Specifying the Bind Address

JBoss provides two ways to specify the bind address: a `bind_addr` XML property and a `bind.address` system property. JBoss uses the `bind.address` system property over the `bind_addr` XML property unless the system property `-Dignore.bind.address` is set to `true`.

You can set clustering parameters, including `bind_addr`, with the `ClusterProtocolStackOption1` parameter in the PolicyCenter `config.xml` file. Set `-Dignore.bind.address=true` when starting JBoss to prevent JBoss from ignoring the `bind_addr` parameter set in `ClusterProtocolStackOption1`.

See “ClusterProtocolStackOption2” on page 47 in the *Configuration Guide*.

Installing a JBoss Instance for Free-text Search

Guidewire free-text search requires that you install a different instance of JBoss than the instance that runs your PolicyCenter application. In a production environment, Guidewire requires that you set up the separate JBoss instance on a host separate from the one that hosts PolicyCenter. This separate instance of the application server runs a full-text search engine, Apache Solr.

Whenever you install a separate JBoss instance for Guidewire free-text search, change the HTTP port to 8983. The standard Solr port is 8983. You configure ports on JBoss in the following file:

`JBOSS_HOME/server/default/conf/bindingservice.beans/META-INF/bindings-jboss-beans.xml`

Edit the file, and change the port property for the WebServer service from 8080 to 8983, as the following example shows.

```
<bean class="org.jboss.services.binding.ServiceBindingMetadata">
  <property name="serviceName">jboss.web:service=WebServer</property>
  <property name="port">8983</property>
```

See also

- “Setting Up Free-text Search for JBoss” on page 93
- “Installing a Tomcat Instance for Free-text Search” on page 24
- “Installing a WebSphere Instance for Free-text Search” on page 26

Configuring Tomcat

This topic provides notes on installing Tomcat to run PolicyCenter. Install Tomcat as a system administrator.

Removing 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.

Do Not Implement 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 the Tomcat Native 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 the encryption 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 the Connector 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 PolicyCenter, so PolicyCenter 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 PolicyCenter 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 PolicyCenter server.

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

Changing Heap Size in Tomcat on Windows

You can increase the Tomcat heap size in one of two ways, depending whether you have installed Tomcat as a Windows service or not.

To increase the Tomcat heap size on Windows when Tomcat is a service

1. Start the Tomcat Windows service.

2. From the Start menu, click **Control Panel**.
3. Click **Administrative Tools**.
4. Click **Services**.
5. Right-click the Tomcat service and click **Properties**.
6. Add the following to **Start parameters**:

```
-Xms1024m -Xmx1024m
```

If you have Tomcat installed, and Tomcat is not run as a Windows Service, set heap size for Tomcat by setting a Windows environment variable.

To increase the Tomcat heap size on Windows when Tomcat is not a service

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 19.

Installing a Tomcat Instance for Free-text Search

Guidewire free-text search requires that you set up a different instance of Tomcat than the instance that runs your PolicyCenter application. In a production environment, Guidewire requires that you set up the separate Tomcat instance on a host separate from the one that hosts your PolicyCenter application. This separate instance of the application server runs a full-text search engine, Apache Solr.

Whenever you install a separate Tomcat instance for Guidewire free-text search, change the port for the HTTP/1.1 protocol to 8983. The standard Solr port is 8983. Edit the file *TOMCAT_HOME/conf/server.xml*, and change the connector port for the HTTP/1.1 protocol from 8080 to 8983, as the following example shows.

```
Connector port="8983" protocol="HTTP/1.1"
      connectionTimeout="20000"
      redirectPort="8443" />
```

See also

- “Setting Up Free-text Search for Tomcat” on page 95
- “Installing a JBoss Instance for Free-text Search” on page 22
- “Installing a Tomcat Instance for Free-text Search” on page 24

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
```

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, PolicyCenter 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 PolicyCenter.

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:

PolicyCenter provides cache coherency (clustering) since it maintains an internal cache of objects for performance reasons. After PolicyCenter changes an object in the cache for a single node in a cluster, PolicyCenter 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 PolicyCenter.

- **Deployment** – WebSphere ND includes tools for automatically deploying configurations to servers in a cluster. Guidewire supports these tools if used in conjunction with our Environment Specific Configuration settings.
- **Load balancing** – WebSphere ND provides load balancing capabilities. See “Load Balancers” on page 21.
- **Clustering** – Clustering is different from the PolicyCenter 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. Maintaining session state information 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 PolicyCenter 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.

Installing a WebSphere Instance for Free-text Search

Guidewire free-text search requires that you install a different instance of WebSphere than the instance that runs your PolicyCenter application. In a production environment, Guidewire requires that you set up the separate WebSphere instance on a host separate from the one that hosts PolicyCenter. This separate instance of the application server runs a full-text search engine, Apache Solr.

Whenever you install a separate WebSphere instance for Guidewire free-text search, change the HTTP port for the default host and its virtual host to 8983. The standard Solr port is 8983. You configure ports on WebSphere through the administrative console.

To change the port number in WebSphere for your default and virtual hosts

1. Start the application server.
2. Change the port for the default host in your WebSphere application server.
 - a. From the Administrative Console, navigate to **Servers** → **Server Types** → **WebSphere application servers** and select your application server from the list or resources that you can administer.
The console displays the configuration page for your application server.
 - b. On the right underneath **Communications**, click **Ports**.
 - c. In the list of TCP/IP ports, click **WC_defaulthost**.
 - d. In the Port field, change the value from 9080 to 8983.
 - e. Click **Apply**.
 - f. In the **Messages** box, Click **Save** to apply the changes to the master configuration.
3. Change the port number for the virtual host in your WebSphere application server.
 - a. From the Administrative Console, navigate to **Environment** → **Virtual hosts** and click the name of your physical host name. The default name of your physical host is `default_host`.
The console displays the configuration page for your physical host.
 - b. On the right underneath **Additional Properties**, click **Host Aliases**.
 - c. Click **New**.

d. Enter the following values.

Field name	Description
Host Name	Enter an asterisk (*).
Port	Enter 8983.

e. Click OK.

f. In the **Messages** box, Click **Save** to apply the changes to the master configuration.

4. Stop and start the application server.

See also

- “Setting Up Free-text Search for WebSphere” on page 97
- “Installing a JBoss Instance for Free-text Search” on page 22
- “Installing a Tomcat Instance for Free-text Search” on page 24

Configuring the Database

Guidewire recommends that you implement the guidelines contained in this topic while you install and configure the database server. PolicyCenter 8.0.4 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 PolicyCenter 8.0.4. 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. PolicyCenter depends heavily on back-end database performance, which in turn depends on storage performance. Optimize the performance of your database server.

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

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

This topic includes:

- “Database Permissions” on page 28
- “Configuring Linguistic Search Collation” on page 28
- “Configuring Compression” on page 28
- “Configuring Oracle for PolicyCenter” on page 32
- “Configuring SQL Server for PolicyCenter” on page 38

Database Permissions

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

Database	Permissions Required
Oracle	<p>The <code>pcUser</code> must have the following permissions on the PolicyCenter database:</p> <ul style="list-style-type: none"> • create session • create procedure • create trigger • create table • create view • create sequence • query rewrite • alter session • select any dictionary <p>If your users want to see statspack data on the PolicyCenter Info Pages interface, grant the <code>pcUser</code> access to the PolicyCenter performance statistics (<code>perfstat</code>) tables.</p>
SQL Server	<p>The <code>pcUser</code> must have the <code>public</code> and <code>db_owner</code> roles on the PolicyCenter database. PolicyCenter supports several different data management pages for performance analysis of the application. To use these pages, the <code>pcUser</code> must be granted <code>view server state</code>. The server login account must also have <code>view database state</code> permission on each PolicyCenter data management view. The data management views all start with <code>sys.dm_</code> prefix.</p>

Configuring Linguistic Search Collation

You can configure how PolicyCenter searches and sorts search results. For example, you can configure whether searching is accent-sensitive or accent-insensitive, case-sensitive or case-insensitive. See “Linguistic Search and Sort” on page 161 in the *Globalization Guide*.

Configuring Compression

Compression reduces the size of the database. The advantage of compression is reduced cost for storage and backups. Compression can also increase performance by effectively making the database buffer caches larger. The disadvantage to compression is that the database requires more CPU time to compress and decompress data. Furthermore, queries that require either a table scan or a full index scan will require fewer physical and logical reads because more rows fit on a single data page.

Consult with Guidewire Support about whether compression will improve overall performance of your PolicyCenter implementation.

The compression settings examples in the following topics demonstrate the syntax to configure compression. These examples are not provided as guidelines for compression settings for your environment.

A full discussion of table and index compression is beyond the scope of this document. Refer to documentation from your database vendor for details about compression options.

Configuring Compression for Oracle

You can specify Oracle compression options for PolicyCenter using the `<ora-compression>` element in `database-config.xml`. The `<ora-compression>` element is nested as shown below:

```
<database>
...
<upgrade>
  <ora-db-ddl>
    <ora-compression table-compression="NONE|BASIC|ADVANCED" index-compression="true|false">
  </ora-db-ddl>
```

```

    </upgrade>
  </database>

```

The `<ora-compression>` element accepts the attributes `table-compression` and `index-compression`. You can specify one or both attributes. Attributes that you specify for `<ora-compression>` apply to all tables and indexes in the database.

You can override options for a specific table by adding an `<ora-table-compression>` element and setting the `table-compression` attribute. The `<ora-table-compression>` element is contained in an `<ora-table-ddl>` element within the `<ora-db-ddl>` element. For example:

```

<database>
...
<upgrade>
  <ora-db-ddl>
    <ora-table-ddl table-name="pc_tableName">
      <ora-table-compression table-compression="NONE|BASIC|ADVANCED" />
    </ora-table-ddl>
  </ora-db-ddl>
</upgrade>
</database>

```

You can override compression options for all indexes on a specific table by including the `index-compression` attribute on the `<ora-table-compression>` element. For example:

```

<database>
...
<upgrade>
  <ora-db-ddl>
    <ora-table-ddl table-name="pc_tableName">
      <ora-table-compression index-compression="true|false" />
    </ora-table-ddl>
  </ora-db-ddl>
</upgrade>
</database>

```

You can override options for a specific index by adding an `<ora-index-ddl>` element within the `<ora-table-ddl>` element for the table that has the index. For example:

```

<database>
...
<upgrade>
  <ora-db-ddl>
    <ora-table-ddl table-name="pc_tableName">
      <ora-index-ddl index-compression="true|false" key-columns="column1,column2" />
    </ora-table-ddl>
  </ora-db-ddl>
</upgrade>
</database>

```

Oracle Table Compression

You can set the `table-compression` attribute of the `<ora-compression>` element and the `ora-table-compression` attribute of the `<ora-table-ddl>` element to `NONE`, `BASIC` or `ADVANCED`. A value of `NONE` specifies that the database or table is not compressed. A value of `BASIC` specifies that the database or table uses Oracle basic compression. A value of `ADVANCED` specifies that the database or table uses Oracle advanced compression. Oracle advanced compression is part of the Oracle Advanced Compression Option, which requires a separate license. Refer to Oracle documentation for more information about compression.

The following example specifies advanced compression for the entire database and no compression for the `pc_Activity` and `pc_Workflow` tables.

```

<database name="PolicyCenterDatabase" dbtype="oracle" autoupgrade="false">
...
<upgrade>
  <ora-db-ddl>
    <ora-compression table-compression="ADVANCED" />
    <ora-table-ddl table-name="pc_Activity">
      <ora-table-compression table-compression="NONE" />
    </ora-table-ddl>
    <ora-table-ddl table-name="pc_Workflow">
      <ora-table-compression table-compression="NONE" />
    </ora-table-ddl>
  </ora-db-ddl>
</upgrade>
</database>

```

```

    </upgrade>
  </database>

```

Oracle Index Compression

You can set the `index-compression` attribute of the `<ora-compression>` and `<ora-index-ddl>` elements to `true` or `false`. Specify an index by setting the `key-columns` attribute of the `<ora-index-ddl>` element to a comma-delimited list of key columns in order. Specify `DESC` after a column name for descending sort order on that column.

An `index-compression` value of `true` specifies to compress all columns but the last for unique indexes and to compress all columns for non-unique indexes.

The following example specifies the following:

- Index compression for the entire database
- No compression for the `pc_Activity` index that contains key columns `PublicID` and `Retired` in key positions one and two, respectively.
- No compression for any indexes on the `pc_Workflow` table.

```

<database>
...
<upgrade>
  <ora-db-ddl>
    <ora-compression index-compression="true">
      <ora-table-ddl table-name="pc_Activity">
        <ora-index-ddl key-columns="PublicID,Retired" index-compression="false" />
      </ora-table-ddl>
      <ora-table-ddl table-name="pc_Workflow">
        <ora-table-compression index-compression="false" />
      </ora-table-ddl>
    </ora-db-ddl>
  </upgrade>
</database>

```

Oracle spatial indexes are not compressible. If you use the `key-columns` attribute to specify a spatial index to compress, PolicyCenter reports an error. If the index is implied to be compressed by the compression configuration of the database or table, PolicyCenter ignores the compression setting for a spatial index.

Configuring Compression for SQL Server

You must have the Enterprise edition of SQL Server to have the SQL Server compression feature.

You can specify SQL Server compression options for PolicyCenter using the `<mssql-compression>` element in `database-config.xml`. The `<mssql-compression>` element is nested as shown below:

```

<database>
...
<upgrade>
  <mssql-db-ddl>
    <mssql-compression table-compression="NONE|PAGE|ROW" index-compression="NONE|PAGE|ROW">
    </mssql-db-ddl>
  </upgrade>
</database>

```

The `<mssql-compression>` element accepts the attributes `table-compression` and `index-compression`. You can specify one or both attributes. Attributes that you specify for `<mssql-compression>` apply to all tables and indexes in the database.

You can override options for a specific table by adding an `<mssql-table-compression>` element and setting the `table-compression` attribute. The `<mssql-table-compression>` element is contained in an `<mssql-table-ddl>` element within the `<mssql-db-ddl>` element. For example:

```

<database>
...
<upgrade>
  <mssql-db-ddl>
    <mssql-table-ddl table-name="pc_tableName">
      <mssql-table-compression table-compression="NONE|PAGE|ROW" />
    </mssql-table-ddl>
  </mssql-db-ddl>

```

```
</upgrade>
</database>
```

You can override compression options for all indexes on a specific table by including the `index-compression` attribute on the `<mssql-table-compression>` element. For example:

```
<database>
...
<upgrade>
  <mssql-db-ddl>
    <mssql-table-ddl table-name="pc_tableName">
      <mssql-table-compression index-compression="NONE|PAGE|ROW" />
    </mssql-table-ddl>
  </mssql-db-ddl>
</upgrade>
</database>
```

You can override options for a specific index by adding an `<mssql-index-ddl>` element within the `<mssql-table-ddl>` element for the table that has the index. For example:

```
<database>
...
<upgrade>
  <mssql-db-ddl>
    <mssql-table-ddl table-name="pc_tableName">
      <mssql-index-ddl key-columns="column1,column2" index-compression="true|false" />
    </mssql-table-ddl>
  </mssql-db-ddl>
</upgrade>
</database>
```

SQL Server Table Compression

You can set the `table-compression` attribute of the `<mssql-compression>` element and the `mssql-table-compression` attribute of the `<mssql-table-ddl>` element to `NONE`, `PAGE`, or `ROW`. A value of `NONE` specifies that the database or table is not compressed. A value of `PAGE` specifies that the database or table uses page-level compression. Page compression is applied only when the page gets full. For page compression, the following operations happen in the following order:

- Row compression
- Prefix compression
- Dictionary compression

A value of `ROW` specifies that the database or table uses row compression. Row compression drastically reduces the metadata needed for variable-length columns.

Refer to SQL Server documentation for more information about SQL Server compression options.

The following example specifies row table compression for the entire database, page compression for the `pc_Activity` table, and no compression for the `pc_Workflow` table.

```
<database name="PolicyCenterDatabase" dbtype="sqlserver" autoupgrade="false">
...
<upgrade>
  <mssql-db-ddl>
    <mssql-compression table-compression="ROW" />
    <mssql-table-ddl table-name="pc_Activity">
      <mssql-table-compression table-compression="PAGE" />
    </mssql-table-ddl>
    <mssql-table-ddl table-name="pc_Workflow">
      <mssql-table-compression table-compression="NONE" />
    </mssql-table-ddl>
  </mssql-db-ddl>
</upgrade>
</database>
```

SQL Server Index Compression

The compression setting of a table is not automatically applied to its non-clustered indexes. You must configure compression settings for indexes separately or in bulk.

You can set the `index-compression` attribute of the `<mssql-compression>` and `<mssql-index-ddl>` elements to `NONE`, `PAGE`, or `ROW`.

Specify an index by setting the `key-columns` attribute of the `<mssql-index-ddl>` element to a comma-delimited list of key columns in order. Specify `DESC` after a column name for descending sort order on that column.

The following example specifies the following:

- Row index compression for the entire database.
- No compression for the `pc_Activity` index that contains key columns `PublicID` and `Retired` in key positions one and two, respectively.
- Page compression for any indexes on the `pc_Workflow` table.

```
<database>
...
<upgrade>
  <mssql-db-ddl>
    <mssql-compression index-compression="ROW">
      <mssql-table-ddl table-name="pc_Activity">
        <mssql-index-ddl key-columns="PublicID,Retired" index-compression="NONE" />
      </mssql-table-ddl>
      <mssql-table-ddl table-name="pc_Workflow">
        <mssql-table-compression index-compression="PAGE" />
      </mssql-table-ddl>
    </mssql-db-ddl>
  </upgrade>
</database>
```

PolicyCenter uses a clustered primary key index. Since this index is actually the table data itself, PolicyCenter uses the compression setting for the table for the primary key index.

Configuring Oracle for PolicyCenter

These directions assume you have already installed Oracle. See the *Guidewire Platform Support Matrix* for information about which specific Oracle versions Guidewire supports for PolicyCenter 8.0.4. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at <https://guidewire.custhelp.com/app/resources/products/platform>.

PolicyCenter requires Oracle Locator, which is included with Oracle Multimedia. Refer to Oracle MOS (My Oracle Support) note *How To Verify That Oracle Locator Is Installed (Doc ID 357943.1)*.

This topic includes:

- “Prerequisites to Installing on Oracle” on page 32
- “Using Oracle Resource Consumer Groups for Slow Policy Queries” on page 34
- “Configuring PolicyCenter to Use Oracle SecureFile LOBs” on page 35
- “Configuring Table Partitioning for Oracle” on page 35
- “Configuring Index Partitioning for Oracle” on page 36
- “Configuring Oracle Date Interval Partitioning” on page 37
- “Configuring Oracle Adaptive Optimization for PolicyCenter” on page 37

Prerequisites to Installing on Oracle

Create a separate user and schema for each Guidewire application. Sharing a schema can result in naming conflicts for primary key constraints and indexes.

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

Oracle Java Virtual Machine (JVM) must be installed on all Oracle databases hosting PolicyCenter. The only exception is when the PolicyCenter application locale is English and you only require case-insensitive searches. Ensure that Oracle initialization parameter `java_pool_size` is set to a value of above 50 MB.

Configure Oracle to use asynchronous I/Os. Asynchronous I/O significantly simplifies a database's I/O 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 PolicyCenter performance. The size of the Oracle database cache is critical to supporting server-side caching. 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 PolicyCenter. There are many tools available for optimizing database performance, including an open source tool, IOzone (www.iozone.org).

To prepare an Oracle database for PolicyCenter

1. Create a new database instance for PolicyCenter.

Guidewire recommends that you not share the PolicyCenter 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 PolicyCenter. 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 PolicyCenter 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 PolicyCenter data tables.
TYPELIST	Stores system code tables.
INDEX	Stores system indexes.
STAGING	Stores inbound staging data tables.
LOB	Stores off-row LOB (large object) data. The LOB tablespace is optional. If you do not specify a physical tablespace for the LOB logical tablespace, then LOB data is stored in the tablespace mapped to the OP logical tablespace. PolicyCenter uses the LOB tablespace for new tables only. For an existing configuration in which the PolicyCenter schema has been created, if you designate an LOB tablespace, PolicyCenter does not move existing LOB columns to the LOB tablespace. If you add an LOB column to an existing table, PolicyCenter does not put the column in the LOB tablespace. If you define a new table with LOB data, PolicyCenter stores the LOB data in the designated LOB tablespace. See "Configuring PolicyCenter to Use Oracle SecureFile LOBs" on page 35.

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

For a development environment, use the same tablespace names as the logical names. The `gwpc dev-dropdb` command only works if the tablespace names match the default logical names.

3. Create a single database user, `pcUser`, in the PolicyCenter database.

4. Grant `pcUser` 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 PolicyCenter Info Pages interface, you also need to grant the `pcUser` access to Statspack's (perfstat user) tables.

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

6. Set default tablespace for `pcUser` 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 PolicyCenter files, Guidewire recommends that you edit your `PATH` variable and move the Oracle directories to the end, after the PolicyCenter 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 Policy Queries

Note: The information in this topic is provided for implementations that experience slow policy queries. If you do not experience slow policy queries, you do not need to define a resource consumer group for policy 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 PolicyCenter to switch to a resource consumer group that you define to perform policy searches. You can set this resource consumer group to have a time limit and cancel SQL operations for policy searches that exceed the time limit.

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

The requirements for using Oracle resource consumer groups for policy queries are:

- The `<oracle-settings>` attribute `db-resource-mgr-cancel-sql` is set in `database-config.xml` to a resource consumer group defined in Oracle. For example:


```
<database name="ClaimCenterDatabase" dbtype="oracle" autoupgrade="false">
  ...
  <oracle-settings db-resource-mgr-cancel-sql="" />
  ...
</database>
```
- The Oracle resource manager plan is set at the system level.
- The `pcUser` has privileges to switch between the two resource groups.

PolicyCenter checks these conditions when the server starts.

Configuring PolicyCenter to Use Oracle SecureFile LOBs

PolicyCenter supports Oracle SecureFile LOBs for unstructured data. To configure PolicyCenter to use SecureFile LOBs, modify the `<database>` block in `database-config.xml`. You can specify to use SecureFile LOBs for all LOBs in the database or for specific tables. You can also configure whether to use caching and the LOB type. The LOB type can be BASIC, SECURE, or SECURE_COMPRESSED. If not specified otherwise PolicyCenter uses SecureFile LOBs.

To specify to use basic file LOBs for all LOBs in the database, add the following to the `<database>` block.

```
<ora-db-ddl>
  <ora-lob type="BASIC" caching="true|false"/>
</ora-db-ddl>
```

To specify to use compressed SecureFile LOBs for all LOBs in the database, add the following to the `<database>` block.

```
<ora-db-ddl>
  <ora-lob type="SECURE_COMPRESSED" caching="true|false"/>
</ora-db-ddl>
```

To specify to use basic file LOBs for all LOBs on a particular table, add the `<ora-lob>` element within the `<ora-table-ddl>` block for the table.

```
<ora-db-ddl>
  <ora-table-ddl name="pc_tablename">
    <ora-lob type="BASIC" />
  </ora-table-ddl>
</ora-db-ddl>
```

If any LOBs are configured to be SecureFile LOBs, and the LOB tablespace is configured, it must be managed with Automatic Segment Space Management. If the LOB tablespace is not configured, then the ADMIN, OP and STAGING tablespaces must be managed with Automatic Segment Space Management.

To specify to use caching for LOBs, add the attribute `caching="true"` to the `<ora-lob>` element.

If you configure a LOB type for the database or for a specific table, you cannot change the configuration after the database has been created. You also cannot change the caching configuration after the database has been created.

Refer to Oracle documentation for information about basic file, SecureFile, and compressed SecureFile LOBs.

Configuring Table Partitioning for Oracle

Table hash partitioning can improve performance of queries on large tables. To enable hash partitioning on a table, add the `<ora-table-hash-partitioning>` element to the `<ora-table-ddl>` block of `<ora-db-ddl>` in `database-config.xml`. For example:

```
<database name="PolicyCenterDatabase" dbtype="oracle" autoupgrade="false">
  ...
  <upgrade>
    ...
    <ora-db-ddl>
      <ora-table-ddl name="Table Name">
        <ora-table-hash-partitioning hash-column="column name" num-partitions="number"/>
      </ora-table-ddl>
    </ora-db-ddl>
  </upgrade>
</database>
```

If a keyable table is partitioned, by default, PolicyCenter uses the ID column as the hash column. You can specify a different column by using the `hash-column` attribute on `<ora-table-hash-partitioning>`. For non-keyable tables, the `hash-column` attribute is required.

If a keyable table is partitioned, PolicyCenter also partitions the primary key index and the index on `PublicID`. This index is on `PublicID` and `Retired` if the table is for a retireable entity.

By default, PolicyCenter uses 128 partitions. You can override this number by defining a `num-partitions` attribute on `<ora-table-hash-partitioning>`.

Note: PolicyCenter creates partitions only when creating a table or index. PolicyCenter does not modify existing tables or indexes. If a table is dropped and rebuilt during an upgrade, PolicyCenter partitions the table if the table is configured to be partitioned. The schema verifier detects and flags if a table is configured as partitioned but is not, or if it is not configured as partitioned but is partitioned.

Configuring Index Partitioning for Oracle

Index partitioning can improve performance of queries in large tables. You can use the DDL configuration element `<ora-index-partitioning>` in `database-config.xml` to specify Oracle index partitioning.

The `<ora-index-partitioning>` XML element is a subelement of `<ora-index-ddl>`, which is itself a subelement of `<ora-table-ddl>`. For example:

```
<database name="PolicyCenterDatabase" dbtype="oracle" autoupgrade="false">
  ...
  <upgrade>
    ...
    <ora-db-ddl>
      <ora-table-ddl name="Table Name">
        <ora-index-ddl key-columns="column1,column2,...">
          <ora-index-partitioning
            partitioning-type="LOCAL|HASH|RANGE"
            // The next two elements apply only to the RANGE partitioning type.
            range-partitioning-column-list="column1,column2,...">
            <ora-index-range-partition value-list=
              "number1|'string1',number2|'string2',..." />
            <ora-index-range-partition value-list=
              "number1a|'string1a',number2a|'string2a',..." />
          ...
        </ora-index-partitioning>
      </ora-index-ddl>
    </ora-table-ddl>
  </ora-db-ddl>
</upgrade>
</database>
```

- The attribute `partitioning-type` is required. This attribute can take the values LOCAL, HASH, or RANGE.
 - LOCAL – The attribute `partitioning-type` is the only attribute allowed, and the index will be partitioned as the table is partitioned.
 - HASH – The index will be globally hash-partitioned on the leading key of the index by using the number of partitions specified or the default number 128.
 - RANGE – The index will be range-partitioned by using the `range-partitioning-column-list` columns and the values specified in the `ora-index-range-partition` elements under this element.

The `range-partitioning-column-list` element takes a comma-delimited list of columns to use for range-partitioning this index. This element requires the definition of one or more `ora-index-range-partition` elements.

The `ora-index-range-partition` defines the value range for each partition in `value-list`. It defines a comma-delimited, ordered list of literal values corresponding to the column list defined in `range-partitioning-column-list`. Any single String value must be inside single quotation marks. The entire list of values must be surrounded by double quotation marks. The values defined are used in the SQL clause `VALUES LESS THAN(value_list)`. Do not specify the last range, which will always be `VALUES LESS THAN (MAXVALUE[, MAXVALUE, ...])`.

Note: PolicyCenter does not support indexes that are range-partitioned on a date column.

For example, the following database block defines an index range partitioning that uses five partitions and two column values per partition. The final partition, created automatically by PolicyCenter, uses the following values for the two columns defined in `range-partitioning-column-list`:

- ADDRESSBOOKUID – At least 'ab:830' and less than MAXVALUE

- RETIRED – At least 0 and less than MAXVALUE

```
<database name="pcDatabase" dbtype="oracle" autoupgrade="false">
...
  <upgrade degree-parallel-ddl="1" verifyschema="true">
    <ora-db-ddl>
      <tablespaces admin="pc_ADMIN" index="pc_INDEX" op="pc_OP"
        staging="pc_STAGING" typelist="pc_TYPELIST"/>
      <ora-table-ddl table-name="pc_CLAIM">
        <ora-index-ddl key-columns="ADDRESSBOOKUID, RETIRED, SUBTYPE, ID">
          <ora-index-partitioning partitioning-type="RANGE"
            range-partitioning-column-list="ADDRESSBOOKUID, RETIRED">
            <ora-index-range-partition value-list="'ab:20', 0"/>
            <ora-index-range-partition value-list="'ab:40', 0"/>
            <ora-index-range-partition value-list="'ab:60', 0"/>
            <ora-index-range-partition value-list="'ab:830', 0"/>
          </ora-index-partitioning>
        </ora-index-ddl>
      </ora-table-ddl>
    </ora-db-ddl>
  </upgrade>
</database>
```

Configuring Oracle Date Interval Partitioning

You can configure PolicyCenter to use Oracle partitioning by date intervals.

To configure date interval partitioning on a table, add an `<ora-table-date-interval-partitioning>` element within `<ora-table-ddl>`. The `<ora-table-ddl>` element is a subelement of `<ora-table-ddl>` in the `<database>` element of `database-config.xml`. The `<ora-table-date-interval-partitioning>` element has two attributes. Use the `datecolumn` attribute to specify a non-nullable timestamp column that PolicyCenter will use to determine partition boundaries. Use the `interval` attribute to specify the period of time for each partition. You can set `interval` to DAILY, WEEKLY, MONTHLY, QUARTERLY, or YEARLY. For example:

```
<database>
...
  <ora-db-ddl>
    <ora-table-ddl table-name="pc_table">
      <ora-table-date-interval-partitioning datecolumn="updateTime" interval="MONTHLY">
    </ora-table-ddl>
  </ora-db-ddl>
</database>
```

PolicyCenter stores partitioned data in the operational tablespace.

Configuring Oracle Adaptive Optimization for PolicyCenter

Oracle 12c introduced a feature called adaptive optimization. Guidewire has conducted performance testing of PolicyCenter with adaptive optimization enabled and has not observed significant performance improvement. Therefore, you might want to disable adaptive optimization for PolicyCenter.

Although you can completely disable adaptive optimization at the database level, this might impact your integrations and non-Guidewire schemas. So Guidewire provides a configuration option to control adaptive optimization for PolicyCenter only.

You can disable adaptive optimization for PolicyCenter by configuring the `adaptive-optimization` attribute of the `<oracle-settings>` element. Or you can set adaptive optimization for PolicyCenter to reporting-only mode. In reporting-only mode, Oracle collects information required for adaptive optimization, but does not modify the plan. You can view this information in the adaptive plan report.

You can set `adaptive-optimization` to the following values:

- REPORTING_ONLY – PolicyCenter sets both `OPTIMIZER_ADAPTIVE_FEATURES` and `OPTIMIZER_ADAPTIVE_REPORTING_ONLY` to TRUE every time it initializes a connection.
- OFF – PolicyCenter sets both `OPTIMIZER_ADAPTIVE_FEATURES` and `OPTIMIZER_ADAPTIVE_REPORTING_ONLY` to FALSE every time it initializes a connection.

For example:

```
<database>
...
  <oracle-settings adaptive-optimization="OFF" />
...
</database>
```

If you do not specify a value for `adaptive-optimization`, PolicyCenter does not set the `OPTIMIZER_ADAPTIVE_FEATURES` and `OPTIMIZER_ADAPTIVE_REPORTING_ONLY` Oracle parameters.

Configuring SQL Server for PolicyCenter

This topic includes information on how to configure SQL Server for PolicyCenter. See the *Guidewire Platform Support Matrix* for information about which specific SQL Server versions Guidewire supports for PolicyCenter 8.0.4. 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.

This topic includes:

- “Prerequisites to Installing on SQL Server” on page 38
- “Configuring SQL Server in Management Studio” on page 39
- “Creating a PolicyCenter Database in SQL Server” on page 39

Prerequisites to Installing on SQL Server

Create a separate database for each Guidewire application. Sharing a database can result in naming conflicts for primary key constraints and indexes.

PolicyCenter requires that the collation of the SQL Server database specifies `CI`, or case-insensitive. The case-sensitivity setting affects table and column names, and PolicyCenter requires these names to be case-insensitive. During startup, PolicyCenter 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 Japanese or Chinese, then the `<sqlserver-settings>` subelement of the `<database>` element in `database-config.xml` must have the `unicodetables` attribute set 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.

Creating a SQL Server database with files of sufficient size and parameters is important to future performance and maintenance. A basic discussion 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 PolicyCenter with other data or applications.

To install PolicyCenter on SQL Server

1. Configure SQL Server. See “Configuring SQL Server in Management Studio” on page 39.
2. Create a database for PolicyCenter. See “Creating a PolicyCenter Database in SQL Server” on page 39.
3. Modify the database-config.xml file so that the application correctly points to the database. See “Deploying PolicyCenter to the Application Server” on page 84.
4. Restart the application server and test by opening PolicyCenter in a browser window.

Configuring SQL Server in Management Studio

To configure SQL Server to support PolicyCenter

1. Open SQL Server Management Studio.
2. In the **Object Explorer**, right-click the server node you plan to use for PolicyCenter 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 PolicyCenter 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](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 PolicyCenter Database in SQL Server

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

IMPORTANT If you plan to create additional database instances to support multiple PolicyCenter 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 PolicyCenter

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 PolicyCenter 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 PolicyCenter logical tablespaces from the **Filegroups** page. If you choose to use filegroups, create a separate filegroup for each logical tablespace:

Logical Name	Usage
ADMIN	Stores system parameters.
OP	Stores the main PolicyCenter data tables.
TYPELIST	Stores system code tables.
INDEX	Stores system indexes.
STAGING	Stores inbound staging data tables.
LOB	Stores off-row LOB (large object) data. The LOB filegroup is optional. If you do not specify a filegroup for the LOB logical tablespace, then LOB data is stored in the filegroup mapped to the OP logical tablespace. PolicyCenter uses the LOB filegroup for new tables only. For an existing configuration in which the PolicyCenter schema has been created, if you designate an LOB filegroup, PolicyCenter does not move existing LOB columns to the LOB filegroup. If you add an LOB column to an existing table, PolicyCenter does not put the column in the LOB filegroup. If you define a new table with LOB data, PolicyCenter stores the LOB data in the designated LOB filegroup.

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 PolicyCenter database connection, you can map the filegroups you created to the PolicyCenter internal logical tablespaces. See “Specifying Filegroups for SQL Server” on page 69.

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, PolicyCenter 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 `pcUser` for the **User name**.

17. Enter the **Login name** that you created earlier.
18. Grant pcUser ownership of the PolicyCenter database by selecting db_owner in both **Schemes owned by this user** and **Database role membership** panels.
19. Click **OK**.
20. PolicyCenter supports several different data management pages for performance analysis of the application. To use these pages, the pcUser must be granted view server state and view database state on each PolicyCenter 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 pcUser.
 - 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 pcUser.
 - i. Select the checkbox to grant view server state permission.
 - j. Select the checkbox to grant create any database permission. This permission allows the gwpc dev-dropdb command to recreate the database.
 - k. 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. PolicyCenter 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, PolicyCenter 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.

Configuring Index Partitioning for SQL Server

Index partitioning can improve performance of queries in large tables. The `partition-scheme` attribute of the `<mssql-index-ddl>` element defines the partition scheme to use for the index. For example:

```
<database name="PolicyCenterDatabase" dbtype="oracle" autoupgrade="false">
  ...
  <upgrade>
    ...
    <mssql-db-ddl>
      <mssql-table-ddl name="Table Name">
        <mssql-index-ddl partition-scheme="partition scheme" />
      </mssql-table-ddl>
    </mssql-db-ddl>
  </upgrade>
</database>
```

Define the partition scheme before starting PolicyCenter with the partition scheme attribute set. The referenced partition scheme in the configuration must exist, or PolicyCenter reports a configuration error during startup.

When a SQL Server index is partitioned, that index is the clustering index for the table. Without a partition scheme defined, the clustering index for a PolicyCenter table is the primary key index.

The partition scheme is treated as a filegroup during index creation, and the SQL Server data space system catalog reports it almost the same as a filegroup.

The PolicyCenter database schema verifier checks that an index and the associated table are stored in the partition scheme configured in `database-config.xml`.

Refer to Microsoft documentation for information about how to create SQL Server partition schemes.

Before Continuing...

Check that your SQL server environment is correct. Test that you can connect to the database using the `pcUser` 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 PolicyCenter 8.0.4. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at <https://guidewire.custhelp.com/app/resources/products/platform>.

Client Information

PolicyCenter is a web application accessed through a web browser. See the *Guidewire Platform Support Matrix* for client system requirements for PolicyCenter 8.0.4. The *Guidewire Platform Support Matrix* is available from the Guidewire Resource Portal at <https://guidewire.custhelp.com/app/resources/products/platform>.

Guidewire has graded levels of support for web browsers. The grades of support are:

Full: Guidewire successfully completed functional testing and performance tuning for the browser and fully supports it.

Partial: Guidewire completed some testing for the browser and found significant functional or performance issues that Guidewire could not resolve. The browser might have a known issue that the browser vendor has not committed to resolving. Browsers with partial support also include browsers that have not yet made it past quality assurance but that Guidewire determines might be fine to use. For example, a newer version of a fully supported browser could be partially supported until Guidewire can fully test the new version. The partial support grade functions largely as a staging area while Guidewire verifies all facets of support.

Unsupported: These are browsers that have significant functional or performance issues. Any browser that does not fully support HTML5 and CSS3 is unsupported by Guidewire. Other less commonly used browsers, such as Opera or Dolphin, could be unsupported because Guidewire has not tested them due to a lack of demand. The unsupported grade functions as a collection of browsers that are unlikely to get any support in the medium to long term.

Enabling DOM Storage

To preserve PolicyCenter user preferences between browser sessions, DOM storage must be enabled in Internet Explorer. DOM storage is enabled by default. If DOM storage is disabled, after a user logs in PolicyCenter opens a popup window with the following message:

Browser DOM Storage disabled: User preferences will not be persistent after page refresh in this browser version.

To enable DOM storage

1. In Internet Explorer, press Alt to open the menu.
2. Click Tools > Internet options.
3. Click Advanced.
4. Under Security, select the Enable DOM Storage checkbox.
5. Click OK.

Installing Java

The PolicyCenter 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 server 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 the Dynamic Code Evolution Virtual Machine

The Dynamic Code Evolution Virtual Machine (DCE VM) is a modified version of the Java HotSpot Virtual Machine (VM). The DCE VM supports any redefinition of loaded classes at runtime. You can add and remove fields and methods and make changes to the super types of a class using the DCE VM. The DCE VM is an improvement to the HotSpot VM, which only supports updates to method bodies.

Guidewire strongly recommends the use of the DCE VM for development in the QuickStart environment. Guidewire does not support the DCE VM for other application servers or in a production environment. Performance of the Java Virtual Machine (JVM) might be impacted by the addition of the DCE VM.

The DCE VM is packaged as an executable JAR file. You can download the DCE VM JAR file and find instructions to install it at:

<https://guidewire.hivelive.com/hives/7a7bd9d656/summary>

Shut down the JVM before you run the DCE VM installer.

To install the DCE VM

1. Download the DCE VM installer.
2. Click Windows button.
3. In the text box, type `cmd` but do not press Enter.
4. Right-click `cmd.exe` and click **Run as administrator**.
5. In the command window, navigate to the directory where you downloaded the DCE VM installer.
6. Enter the following command:

```
java -jar dcevm-installer.jar
```
7. On the installer window, specify the JDK that you are using for development of PolicyCenter.
8. Click Install. The installer replaces `%JAVA_HOME%/jre/bin/client/jvm.dll` and `%JAVA_HOME%/jre/bin/server/jvm.dll`. The installer creates copies of the original `jvm.dll` files and names these files `jvm.dll.backup`. The installer also adds `dcevm.jar` to `%JAVA_HOME%/jre/lib/ext`.

Once you have installed the DCE VM, you can confirm the installation by running the `java -version` command. The output from the `java -version` command lists the Java HotSpot 64-bit Server VM after the Java version information.

See also

- “Studio and the DCE VM” on page 92 in the *Configuration Guide*
- <http://java.net/projects/dcevm/>

Installing Ant

Ant is a common tool used for platform-independent scripting with Java systems. The PolicyCenter 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 an ANT_OPTIONS environment variable set, do not include the -XX:+UseParallelGC option. The -XX:+UseParallelGC option is not compatible with the Dynamic Code Evolution Virtual Machine (DCE VM), which requires use of its embedded serial garbage collector. Any -XX options are not standard options. These options can vary between platforms and releases and some -XX options might be incompatible with the DCE VM. If you have issues running Guidewire gwpc commands, remove any -XX options from ANT_OPTIONS 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 PolicyCenter can locate them. Make these environment variables available in the user environment in which you plan to run PolicyCenter. 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\jdk1.7.0_17 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 20 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	C:\Program Files\Java\jdk1.7.0_17
	ANT_HOME	C:\ant
	Add Ant to Path	C:\ant\bin

Before Continuing

Check that you established your environment correctly. Open a new command prompt 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 PolicyCenter. Enter your configuration values in the following tables:

Database Configuration	Value
Database Name	
Server Name	
Database server port	
PolicyCenter 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	

Installing a PolicyCenter Development Environment

PolicyCenter 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 48.
- Configure Tomcat to automatically load configuration changes that you make in Guidewire Studio. For instructions, see “Installing a Tomcat Development Environment” on page 51.
- 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 84
 - “Installing a WebSphere Production Environment” on page 87
 - “Installing a WebLogic Production Environment” on page 85

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

IMPORTANT This topic only provides information for installing a PolicyCenter development environment. To install a production environment, first review “Preparing a PolicyCenter Environment” on page 15 and then proceed to “Installing a PolicyCenter Production Environment” on page 59.

This topic includes:

- “Using Multiple PolicyCenter Development Instances” on page 48
- “Installing the QuickStart Development Environment” on page 48
- “Installing a Tomcat Development Environment” on page 51
- “Using the QuickStart Database” on page 53

- “Using SQL Server or Oracle in a Development Environment” on page 54
- “Configuring Archiving for Development Testing” on page 54
- “Enabling Reinsurance Management or Disabling Work Queue” on page 55
- “Installing Sample Data” on page 55

Using Multiple PolicyCenter Development Instances

Occasionally, a developer might want to connect from one machine to multiple PolicyCenter 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 a session cookie with the same host name and path. The 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. 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 PolicyCenter 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 48
- “QuickStart Development Environment Prerequisites” on page 49
- “Installing with QuickStart” on page 49
- “QuickStart Commands” on page 50
- “QuickStart Application Server” on page 50
- “Troubleshooting QuickStart” on page 51

Advantages to Using the QuickStart Software

The bundled lightweight application server and database provided with PolicyCenter 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 PolicyCenter rapidly without any configuration.
- Configure PolicyCenter 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 PolicyCenter.
- Make changes to PolicyCenter using Guidewire Studio.
- Make changes to the product model and view the output.

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 PolicyCenter configuration files from the file system, rather than requiring a packaged WAR or EAR file. Therefore, developers can configure PolicyCenter without needing to repack 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 51. 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>.
- PolicyCenter requires Java with the Dynamic Code Evolution Virtual Machine (DCE VM) and Ant. 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 43 and “Installing Ant” on page 45 for installation guidelines.

Installing with QuickStart

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

PolicyCenter uses a QuickStart database by default. See “Using the QuickStart Database” on page 53. You can configure a PolicyCenter development environment to use an Oracle or SQL Server database instead. See “Using SQL Server or Oracle in a Development Environment” on page 54 for instructions.

For instructions to install a QuickStart ContactManager development environment, see “Installing ContactManager with QuickStart for Development” on page 43 in the *Contact Management Guide*.

To install PolicyCenter with the bundled QuickStart application server

1. Create an installation directory for PolicyCenter. This guide uses `PolicyCenter` as the directory name. Do not use spaces in the installation directory path. Studio will not run from a directory with a space in its name.
2. Unzip the PolicyCenter ZIP file into the PolicyCenter directory. See “Setting Font Display Options” on page 115 in the *Configuration Guide* for a list of how the files are organized.
3. Open a command prompt to `PolicyCenter/bin`.
4. If you are reinstalling PolicyCenter and using a QuickStart database, drop the QuickStart database by running `gwpc dev-dropdb`.
5. PolicyCenter uses the QuickStart database by default. You can configure where PolicyCenter stores the QuickStart database files. See “Using the QuickStart Database” on page 53. 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 54. Then continue with this procedure after you have created the database account and configured PolicyCenter to connect to the database.
6. Start the QuickStart server with the following command:

```
gwpc dev-start
```

When the server has started, you see: `*****PolicyCenter ready*****` in the command window.
7. Open a browser and navigate to `http://localhost:8180/pc` and login with the default superuser.
 - User name is `su`.
 - Password is `gw`.

After installing PolicyCenter 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 53
- “Using SQL Server or Oracle in a Development Environment” on page 54
- “Enabling Reinsurance Management or Disabling Work Queue” on page 55
- “Installing Sample Data” on page 55
- “Configuring Logging” on page 23 in the *System Administration Guide*
- To integrate PolicyCenter with ContactManager, see “Integrating ContactManager with Guidewire Core Applications” on page 49 in the *Contact Management Guide*.

QuickStart Commands

You launch many PolicyCenter commands by passing arguments to the `gwpc` command, located in `PolicyCenter/bin`. For a complete list of `gwpc` commands, see “Commands Reference” on page 119.

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. The QuickStart application server is suitable for demonstration and development environments. The QuickStart application server is not suitable nor supported for production environments.

PolicyCenter on the QuickStart server always runs in development mode. You cannot run PolicyCenter on the QuickStart server in production mode.

See also

“Server Modes and Run Levels” on page 58 in the *System Administration Guide*.

Configuring QuickStart Ports

The `PolicyCenter/modules/configuration/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 PolicyCenter QuickStart application server listens on port 8180 by default.

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

Tuning QuickStart Application Server Memory

The `PolicyCenter/modules/configuration/etc/memory.properties` file specifies memory settings for the QuickStart application server and other `gwpc` 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, edit the following values in `PolicyCenter/modules/configuration/etc/memory.properties`:

- Starting heap size – `com.guidewire.commons.jetty.GWServerJettyServerMain.xmls`
- Maximum heap size – `com.guidewire.commons.jetty.GWServerJettyServerMain.xmlx`
- Maximum permanent size – `com.guidewire.commons.jetty.GWServerJettyServerMain.maxperm`

You can also adjust the memory settings for other `gwpc` tools.

See also

“Tuning Command Line Tool Memory Settings” on page 119.

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 prompt, navigating to PolicyCenter/bin and entering the following command:

```
gwpc dev-stop
```

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

Issue: You experience port conflicts.

Solution: The QuickStart server listens on a default server port. The default server port might already be in use by your organization. Consult with your IT department to verify which ports to use.

See also

“Configuring QuickStart Ports” on page 50.

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 PolicyCenter development environment on Tomcat in which you can make configuration changes without having to repackage the application.

To configure a Tomcat ContactManager development environment, see “Installing ContactManager with Tomcat and SQL Server for Development” on page 45 in the *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 `ClassNotFoundException`s and other problems and prevents the application from running properly.

See also “Using Multiple PolicyCenter Development Instances” on page 48.

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 42.
- PolicyCenter requires Java and Ant. See “Installing Java” on page 43 and “Installing Ant” on page 45 for installation guidelines.
- Do not include spaces in the path to your Tomcat installation.

Building a PolicyCenter Development Environment with Tomcat

The following procedure installs a PolicyCenter development environment on Tomcat. If you want to set up a production environment on Tomcat, see “Installing a Tomcat Production Environment” on page 84

To set up a PolicyCenter development environment on Tomcat

1. If you have not already done so, create an installation directory for PolicyCenter. This guide uses PolicyCenter as the directory name.
2. Unzip the PolicyCenter ZIP file into the PolicyCenter directory. See “Setting Font Display Options” on page 115 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 PolicyCenter directory. This enables you to recover quickly from accidental changes that can prevent PolicyCenter 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 prompt to PolicyCenter/bin.
6. If you are reinstalling PolicyCenter and using the QuickStart database, drop the database by running `gwpc dev-dropdb`.
7. PolicyCenter uses the QuickStart database by default. You can configure where PolicyCenter stores the QuickStart database files. See “Using the QuickStart Database” on page 53. 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 54. Then continue with this procedure after you have created the database account and configured PolicyCenter to connect to the database.
8. Run the following command:
`gwpc build-tomcat-war-dbc`
 The `gwpc build-tomcat-war-dbc` command creates a `pc.war` file including JDBC drivers and places the WAR file in the PolicyCenter/dist/war directory.
9. Deploy the package to Tomcat by copying the `pc.war` file to the webapps directory in your Tomcat server.
10. Use the Tomcat `bin/startup.bat` command to start Tomcat and allow it to explode the WAR file. When Tomcat starts, it automatically recognizes the new application and unpacks the `pc.war` into a directory structure within webapps. For this example, Tomcat creates a `webapps/pc` directory. Each time you deploy a new copy of a `pc.war` file, delete the existing `pc` directory structure before you start Tomcat.
11. Delete the Tomcat `webapps\pc\modules\configuration` directory.
12. Create a symbolic link from the deployed PolicyCenter application on Tomcat to the PolicyCenter\modules\configuration directory of the original PolicyCenter installation location.
Windows 7 and Vista: Use the `mklink` command to create the link, as in the example below:
`mklink /d C:\apache-tomcat-7.0.39\webapps\pc\modules\configuration C:\PolicyCenter\modules\configuration`
Windows XP: Use the Windows SysInternals program `Junction.exe` to create the symbolic link. The `Junction.exe` program is available at:
<http://technet.microsoft.com/en-us/sysinternals/bb896768.aspx>
13. Restart the Tomcat server.

To test the PolicyCenter development environment on Tomcat

1. If not already running, start the Tomcat server.
2. Open a browser to the PolicyCenter development environment URL. For example:
`http://localhost:8080/pc`
 Notice that the login page includes a **User name** field. You change the **User name** field in this procedure.
3. Launch Guidewire Studio by running the following command from PolicyCenter/bin:
`gwpc studio`
4. Expand Localizations.

5. Expand the locale you are using.
6. Double-click `display.properties`.
7. Select the **Web.Login.Username** display key.
8. Change the value of the **Web.Login.Username** display key in an obvious manner. For example, add several question marks.
9. Click the Save All icon to save your changes.
10. Log into PolicyCenter with the `su` account. At this point the **User name** field is unchanged.
11. Press ALT+SHIFT+T to open the **Server Tools** page.
12. Select **Internal Tools**.
13. Select **Reload**.
14. Click **Reload PCF Files**.
15. Click **Log Out**. PolicyCenter redirects you to the login page.
16. 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 PolicyCenter 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 53
- “Using SQL Server or Oracle in a Development Environment” on page 54
- “Configuring Archiving for Development Testing” on page 54
- “Enabling Reinsurance Management or Disabling Work Queue” on page 55
- “Installing Sample Data” on page 55
- “Configuring Logging” on page 23 in the *System Administration Guide*
- To integrate PolicyCenter with ContactManager, see “Integrating ContactManager with Guidewire Core Applications” on page 49 in the *Contact Management Guide*.

Using the QuickStart Database

By default, PolicyCenter uses the QuickStart database. The QuickStart database is the *H2 Database Engine*. Guidewire includes the QuickStart database for the convenience of those who need a lightweight solution for demonstration and configuration purposes. By default, PolicyCenter 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 `database-config.xml` file. For example:

```
<!-- H2 (meant for dev/quickstart use only!) -->
<database name="PolicyCenterDatabase" dbtype="h2" autoupgrade="true">
  <dbcp-connection-pool jdbc-url="jdbc:h2:mem:/tmp/guidewire/pc"/>
</database>
```

Guidewire uses `tmp/guidewire` as the database file location and `pc` as the file prefix in the default configuration. You can modify the database file 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 prompt at `PolicyCenter/bin` and enter `gwpc 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 Guidewire Development Environment” on page 85 in the *Configuration Guide*.

Using SQL Server or Oracle in a Development Environment

You can use Oracle or SQL Server with the QuickStart application server instead of the QuickStart database. Such a configuration can be used as a development environment only. Guidewire does not support using the Quickstart server for a production environment.

You can also use Oracle or SQL Server with the Tomcat application server instead of the QuickStart database. Such a configuration can be used as a development or production environment.

For instructions on creating a SQL Server or Oracle database instance, consult “Configuring the Database” on page 27. For instructions on configuring PolicyCenter to connect to the database, consult “Configuring a Database Connection” on page 66.

If you are using the QuickStart application server, use the commands listed in “QuickStart Command Tools” on page 120 to control the application server.

Configuring 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 and restore work queues.

To disable the archive and restore work queues

1. In a command window, navigate to the `PolicyCenter\bin` directory in the PolicyCenter installation.
2. Launch Guidewire Studio using the following command:

```
gwpc studio
```

3. In the **Project** window, navigate to **configuration** → **config** → **workqueue**, and then open `work-queue.xml`.

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

```
<work-queue workQueueClass="com.guidewire.pc.domain.archive.ArchivePolicyTermWorkQueue"
  progressinterval="600000">
  <worker instances="10"/>
</work-queue>
<work-queue workQueueClass="com.guidewire.pc.domain.archive.RestorePolicyTermWorkQueue"
  progressinterval="600000">
  <worker instances="10"/>
</work-queue>
```

5. Save your changes.

See “More Information on Archiving” on page 329 in the *Application Guide*.

Enabling Reinsurance Management or Disabling Work Queue

Guidewire Reinsurance Management is available within PolicyCenter. However, Reinsurance Management is licensed separately from PolicyCenter. Contact your Guidewire sales representative for information on how to obtain Reinsurance Management. Contact your Guidewire support representative for instructions on how to enable Reinsurance Management.

See “Reinsurance Management Concepts” on page 617 in the *Application Guide*.

If you do not enable Reinsurance Management, Guidewire recommends that you disable `RICedingWorkQueue`.

To disable `RICedingWorkQueue`

1. In a command window, navigate to the `PolicyCenter/bin` directory in the PolicyCenter installation.

2. Launch Guidewire Studio using the following command:

```
gwpc studio
```

3. In Studio, open `workqueue` → `work-queue.xml`.

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

```
<work-queue workQueueClass="com.guidewire.pc.domain.reinsurance.RICedingWorkQueue"
  progressinterval="30000">
  <worker instances="1"/>
</work-queue>
```

5. Save your changes.

Installing Sample Data

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

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

The PolicyCenter server must be in development mode to load sample data. The QuickStart server is always in development mode. If you are using a Tomcat server in your development environment, start the server in development mode. See “Building a PolicyCenter Development Environment with Tomcat” on page 51.

If you have already loaded a sample data set, you must drop the database before you load a different sample data set.

To install sample data

1. Log into PolicyCenter with the su account.
2. Press ALT+SHIFT+T to open the **Server Tools** page.
3. Click **Internal Tools**.
4. Click **PC Sample Data** in the menu on the left.
5. Click **Load** for one or more of the following.

- **Tiny** – Provides a small amount of data. Load the **Tiny** data set for unit tests.
- **Free-text Search** – A separate additional data set of accounts and policies for testing and demonstrating Guidewire free-text search.

Guidewire recommends that you set up and enable free-text search before you load free-text sample data. PolicyCenter indexes the sample data automatically if you load the data after you set up and enable free-text search. If you load free-text sample data before you set up and enable free-text search, you then must perform an extra step to index the sample data.

For more information, see “Free-text Search Configuration” on page 357 in the *Configuration Guide*.

- **Small** – Includes all of the **Tiny** set plus a few sample accounts and policies. Load the **Small** data set for local configuration.
- **Large** – Includes all of the **Small** set plus a full set of data. Load the **Large** data set for demonstrations, manual quality assurance, and performance testing.
- **Product x Job Status** – Additional data set containing policies for every product in every job status allowed by GUnit entity builders.

PolicyCenter now contains some sample data.

6. To load sample data for the location search API used by the catastrophe search from ClaimCenter, click **Load catastrophe sample policies (commercial property)**.

To demonstrate the difference in rate calculation between written date and effective date, sample data for all states beginning with the letter "N" are configured to use written date. Rates for those states are chosen based on the current date and not the effective date of the policy. All other states are configured to use the effective date of the policy. This has nothing to do with actual industry practices or state laws and is only intended as a demonstration.

Using Gosu to Configure Sample Data

This topic describes using Gosu to configure the sample data.

About Sample Data Gosu Classes

The simplest way to configure sample data in PolicyCenter is to edit the Gosu in the `gw.sampledata` package.

The contents of a data set are in subpackages of `gw.sampledata` based on the typecode. For example, the contents of the small data set are within `gw.sampledata.small`. The contents are further subdivided into collections by the kind of data. For instance, to alter the Account data in the small sample data set, edit `SmallSampleAccountData.gs`.

A few guidelines for editing the data:

- The main `SampleData` class just invokes the appropriate data collections. Put the data in there.
- When generating data, use the helper methods in `AbstractSampleDataCollection`, creating your own as necessary. Do not put complex logic in the collection itself.
- If you have frequently used constants, consider putting them in `SampleDataConstants` for reuse.

Customizing Gosu Classes for Sample Data

To configure the Gosu classes used to load sample data into PolicyCenter

1. Start Studio and the PolicyCenter server and connect Studio to PolicyCenter.
2. In Studio, expand **configuration** → **Classes** → **gw** → **sampledata** → **tiny** and open `TinySampleCommunityData.gs`.
3. Copy one entry from the `// USER` section and make changes, creating a new user.

```
var aapplegate = loadUser(bundle, "underwriter", "underwriter",
    enigmaOrg, false, false, false, "aapplegate",
    "aapplegate@enigma_fc.com", "Alice", "Applegate",
    "213-555-8164", "143 Lake Ave. Suite 501",
    "Pasadena", "CA", "91253", "US")
```
4. From the File menu, click **Save Changes**.
5. Log into PolicyCenter with the `su` account.
6. Press ALT+SHIFT+T to open the **Server Tools** page.
7. Select **PC Sample Data**.
8. Click **Load** for the **Tiny** dataset.
9. Logout and log back in as the new user to verify the user has been created, indicating your new sample data has loaded correctly.

About Importing and Exporting Data in PolicyCenter

For instructions on how to import or export administrative data, consult “Importing and Exporting Administrative Data” on page 95 in the *System Administration Guide*.

Installing a PolicyCenter Production Environment

Installing a PolicyCenter 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 PolicyCenter to a production environment.

For instructions to install a PolicyCenter development environment, see “Installing a PolicyCenter Development Environment” on page 47 instead of this topic.

This topic includes:

- “Unpacking the Configuration Files” on page 59
- “Configuring a Database Connection” on page 66
- “Deploying PolicyCenter to the Application Server” on page 84

Unpacking the Configuration Files

Guidewire packages PolicyCenter 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 PolicyCenter configuration. These directions assume you plan to maintain the configuration files on the administrative workstation.

To unpack the PolicyCenter configuration environment

1. If you have not already done so, create an installation directory for PolicyCenter. This guide uses `PolicyCenter` as the directory name. Do not use spaces in the installation directory path. Studio will not run from a directory with a space in its name.
2. Unzip the PolicyCenter ZIP file into the PolicyCenter directory.

3. If you are not using a version control system, make a read-only copy of the PolicyCenter directory. This enables you to recover quickly from accidental changes that can prevent PolicyCenter from starting.

At this point, you have a full set of PolicyCenter configuration files. For an overview of the directories included with PolicyCenter, see “Setting Font Display Options” on page 115 in the *Configuration Guide*.

Guidewire recommends that you maintain your PolicyCenter configuration files in a change control system such as Perforce or SVN. If you have such a system, add your PolicyCenter installation directory and files to it at this point.

Key PolicyCenter gwpc Commands

The following key commands are available with PolicyCenter after unpacking the PolicyCenter ZIP file. Launch these commands by passing the command name as a parameter to the gwpc utility in PolicyCenter/bin:

Command	Action
gwpc -p	Displays all gwpc command options.
gwpc build-jboss-war-dbc	<p>Builds the generic WAR file for JBoss including JDBC drivers. Use gwpc build-jboss-war-dbc if you are going to have PolicyCenter manage the database connection pool.</p> <p>You can include the Boolean parameter config.war.dictionary=true to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command:</p> <pre>gwpc build-jboss-war-dbc -Dconfig.war.dictionary=true</pre> <p>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 <i>Data Dictionary</i> and <i>Security Dictionary</i> respectively. To view a dictionary, open index.html in the data or security folder.</p>
gwpc build-jboss-war-jndi	<p>Builds the generic WAR file for JBoss without JDBC drivers. Use gwpc build-jboss-war-jndi only if you are going to use a JNDI database connection managed by JBoss. See "Using a JNDI Data Source" on page 72.</p> <p>You can include the Boolean parameter config.war.dictionary=true to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command:</p> <pre>gwpc build-jboss-war-jndi -Dconfig.war.dictionary=true</pre> <p>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 <i>Data Dictionary</i> and <i>Security Dictionary</i> respectively. To view a dictionary, open index.html in the data or security folder.</p>
gwpc build-tomcat-war-dbc	<p>Builds the generic WAR file for Tomcat including JDBC drivers. Use gwpc build-tomcat-war-dbc if you are going to have PolicyCenter manage the database connection pool.</p> <p>You can include the Boolean parameter config.war.dictionary=true to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command:</p> <pre>gwpc build-tomcat-war-dbc -Dconfig.war.dictionary=true</pre> <p>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 <i>Data Dictionary</i> and <i>Security Dictionary</i> respectively. To view a dictionary, open index.html in the data or security folder.</p>
gwpc build-tomcat-war-jndi	<p>Builds the generic WAR file for Tomcat without JDBC drivers. Use gwpc build-tomcat-war-jndi only if you are going to use a JNDI database connection managed by JBoss. See "Using a JNDI Data Source" on page 72.</p> <p>You can include the Boolean parameter config.war.dictionary=true to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command:</p> <pre>gwpc build-tomcat-war-jndi -Dconfig.war.dictionary=true</pre> <p>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 <i>Data Dictionary</i> and <i>Security Dictionary</i> respectively. To view a dictionary, open index.html in the data or security folder.</p>

Command	Action
<code>gwpc build-weblogic-ear-dbc</code>	Builds the EAR file for WebLogic including JDBC drivers. Use <code>gwpc build-websphere-ear-dbc</code> if you are going to have PolicyCenter manage the database connection pool.
<code>gwpc build-weblogic-ear-jndi</code>	Builds the EAR file for WebLogic without JDBC drivers. Use <code>gwpc build-websphere-ear-jndi</code> only if you are going to use a JNDI database connection managed by WebLogic. See “Using a JNDI Data Source” on page 72.
<code>gwpc build-websphere-ear-dbc</code>	Builds the EAR file for WebSphere including JDBC drivers. Use <code>gwpc build-websphere-ear-dbc</code> if you are going to have PolicyCenter manage the database connection pool.
<code>gwpc build-websphere-ear-jndi</code>	Builds the EAR file for WebSphere without JDBC drivers. Use <code>gwpc build-websphere-ear-jndi</code> only if you are going to use a JNDI database connection managed by WebSphere. See “Using a JNDI Data Source” on page 72.
<code>gwpc displaykey-diff</code>	<p>A display key difference tool that does the following:</p> <ul style="list-style-type: none"> • Compares each locale configured on the server against the master display key list. • Generates a file that contains a list of any missing keys. <p>See “Localizing Typecodes” on page 48 in the <i>Globalization Guide</i>.</p>
<code>gwpc export-ll0ns</code> <code>-Dexport.file="translation_file"</code> <code>-Dexport.locale="language to export"</code>	<p>Exports a translation file from PolicyCenter into a file.</p> <p>The <code>-Dexport.file</code> parameter specifies the destination file.</p> <ul style="list-style-type: none"> • If you leave the import translation file in the same location, then enter only the name of the file to import. • If you move the translation file to a different location, then enter an absolute path or a relative path to the file from the root of the installation directory. <p>The <code>-Dexport.locale</code> parameter specifies the destination language to export. The <code>-Dexport.locale</code> parameter must match a PolicyCenter LanguageType typecode, such as <code>fr</code> or <code>ja</code>.</p> <p>See “Localizing Typecodes” on page 48 in the <i>Globalization Guide</i>.</p>
<code>gwpc import-ll0ns</code> <code>-Dimport.file="translation_file"</code> <code>-Dimport.locale=destination_locale</code>	<p>Imports a translation file into the configuration.</p> <p>The <code>-Dimport.file</code> parameter specifies the file that contains the translations. It must be in the same format as an export file from Studio.</p> <p>The <code>-Dimport.locale</code> parameter specifies the destination language for the translations. The language must match a PolicyCenter LanguageType typecode, such as <code>fr</code> or <code>ja</code>.</p> <p>See “Localizing Typecodes” on page 48 in the <i>Globalization Guide</i>.</p>
<code>gwpc install-localized-module</code> <code>-Dmodule.file=ZipFileName</code> <code>-Dinstall.type={install upgrade}</code>	Installs or upgrades a language module. See “Installing Display Languages” on page 23 in the <i>Globalization Guide</i> .
<code>gwpc iterator-upgrade</code>	Upgrades all iterators on toolbar buttons and filters. This command is only used during upgrade from a prior major version. See “Running PCF Iterator Upgrade” on page 270 in the <i>Upgrade Guide</i> .
<code>gwpc regen-datamapping-split</code>	<p>Builds the data mapping files with files split out by table and typelist. Data mapping files represent fields present in the physical database. None of the virtual fields are represented.</p> <p>Class: <code>com.guidewire.tools.datamapping.DataMappingTool</code></p>
<code>gwpc regen-datamapping-together</code>	<p>Builds the data mapping files with all tables and typelists concatenated. represent fields present in the physical database. None of the virtual fields are represented.</p> <p>Class: <code>com.guidewire.tools.datamapping.DataMappingTool</code></p>

Command	Action
<pre> gwpc regen-dictionary -DmaxSPVInclusions=<i>n</i> -DoutputFormat={html xml} </pre>	<p>Generates the <i>Data Dictionary</i> and <i>Security Dictionary</i>. The <i>Data Dictionary</i> includes physical fields in the database and virtual fields in the data model. The <i>Security Dictionary</i> includes application permission keys, system permissions, and roles.</p> <p>Generate the dictionaries the first time you unzip PolicyCenter and each time you update the data model. Run the <code>gwpc regen-java-api</code> and <code>gwpc regen-soap-api</code> commands each time just prior to regenerating the security and data dictionaries.</p> <p><code>gwpc regen-dictionary</code></p> <p>To view either dictionary in HTML format, open the index file for it in a browser:</p> <pre> PolicyCenter/dictionary/data/index.html PolicyCenter/dictionary/security/index.html </pre> <p>You can generate the Data Dictionary and Security Dictionary in XML format, with associated XSD files. Use the generated XML and XSD files to import the <i>Data Dictionary</i> and <i>Security Dictionary</i> into third-party database design tools.</p> <pre> gwpc regen-dictionary -DoutputFormat=xml </pre> <p>This command generates the following XML and XSD files for the dictionaries:</p> <pre> GenericCenter/build/dictionary/data/entityModel.xml GenericCenter/build/dictionary/data/entityModel.xsd GenericCenter/build/dictionary/security/ securityDictionary.xml GenericCenter/build/dictionary/security/ securityDictionary.xsd </pre> <p>You can generate the dictionaries in HTML format while building a WAR file. See the description for <code>gwpc build-jboss-war</code> or <code>gwpc build-tomcat-war</code> for instructions.</p> <p>For more information, see “Regenerating the Data Dictionary and Security Dictionary” on page 32 in the <i>Configuration Guide</i>.</p> <p>This command performs PCF validation to catch errors in PCF files, such as:</p> <ul style="list-style-type: none"> • Invalid expressions • Attributes that have no meaning when another attribute is set • Editable cells within non-editable objects • Illegal use of check boxes • Invalid arguments • Other errors <p>Server commands that perform PCF validation, including <code>regen-dictionary</code>, <code>regen-pcfmapping</code>, and <code>verify-resources</code> can take a very long time to complete. The server performs a second-pass verification which, among other operations, verifies all possible combinations of modal sections on each PCF page. For example, a page with 12 modes that is used four times causes the server command to validate 12 x 12 x 12 x 12 = 20,736 combinations. To limit the number of shared section verifications performed by these commands, specify the optional parameter <code>maxSPVInclusions</code>. This parameter defines the depth for second pass verification that limits the number of shared sections that are included in the verification of PCF types. For instance:</p> <pre> gwpc regen-dictionary -DmaxSPVInclusions=1000 </pre> <p>In this case, the second pass compilation of PCF files would stop after 1000 permutations of modal PCF files.</p> <p>Experiment with values for <code>maxSPVInclusions</code> between 1000 and 1000000 to achieve improved command completion times. However, be aware that limiting the validation depth means that some combinations of PCF modes and uses are not validated.</p> <p>The <code>maxSPVInclusions</code> property can only be specified as a positive integer value.</p> <p>For more information on second pass verification, see “Setting Verification Options” on page 116 in the <i>Configuration Guide</i>.</p> <p>Classes:</p> <pre> com.guidewire.tools.dictionary.data.DataDictionaryTool com.guidewire.tools.dictionary.security.SecurityDictionaryTool </pre>

Command	Action
<code>gwpc regen-from-wsc</code>	<p>Downloads the WSDL and XSD files for all WSC (Web Service Collection) files. Web service collection files encapsulate the set of resources necessary to connect to a web service on an external system. If you view a web service collection in Studio and click the Fetch Updates button, Studio gets the latest WSDL and XSD files from servers that publish those web services. This tool is equivalent to the Fetch Updates process, but runs from the command line and operates on all web services rather than one.</p> <p>See “Loading WSDL Locally Using Studio Web Service Collections” on page 76 in the <i>Integration Guide</i>.</p>
<code>gwpc regen-gosudoc</code>	<p>Generates Gosu API reference of the APIs available from Gosu within Studio. This command produces documentation at <code>PolicyCenter/build/gosudoc/index.html</code>. See “Gosu Generated Documentation (Gosudoc)” on page 38 in the <i>Gosu Reference Guide</i>.</p> <p>Class: <code>com.guidewire.tools.gosudoc.GosuDocMain</code></p>
<code>gwpc regen-java-api</code>	<p>Builds the Java API libraries to the <code>PolicyCenter/java-api</code> directory. See “Regenerating Integration Libraries and WSDL” on page 31 in the <i>Integration Guide</i>.</p>
<code>gwpc regen-pcfmapping -DmaxSPVInclusions=n</code>	<p>Builds the PCF mappings.</p> <p>This command performs PCF validation to catch errors in PCF files, such as:</p> <ul style="list-style-type: none"> • Invalid expressions • Attributes that have no meaning when another attribute is set • Editable cells within non-editable objects • Illegal use of check boxes • Invalid arguments • Other errors <p>Server commands that perform PCF validation, including <code>regen-dictionary</code>, <code>regen-pcfmapping</code>, and <code>verify-resources</code> can take a very long time to complete. The server performs a second-pass verification which, among other operations, verifies all possible combinations of modal sections on each PCF page. For example, a page with 12 modes that is used four times causes the server command to validate $12 \times 12 \times 12 \times 12 = 20,736$ combinations. To limit the number of shared section verifications performed by these commands, specify the optional parameter <code>maxSPVInclusions</code>. This parameter defines the depth for second pass verification that limits the number of shared sections that are included in the verification of PCF types. For instance:</p> <pre>gwpc regen-pcfmapping -DmaxSPVInclusions=1000</pre> <p>In this case, the second pass compilation of PCF files would stop after 1000 permutations of modal PCF files.</p> <p>Experiment with values for <code>maxSPVInclusions</code> between 1000 and 1000000 to achieve improved command completion times. However, be aware that limiting the validation depth means that some combinations of PCF modes and uses are not validated.</p> <p>The <code>maxSPVInclusions</code> property can only be specified as a positive integer value.</p> <p>For more information on second pass verification, see “Setting Verification Options” on page 116 in the <i>Configuration Guide</i>.</p> <p>Class: <code>com.guidewire.tools.pcfmapping.PCFMappingWriterMain</code></p>
<code>gwpc regen-phone-metadata</code>	<p>Regenerates phone metadata in <code>config/phone/data</code>. Run this command if you have modified the phone metadata XML files</p>
<code>gwpc regen-rulereport</code>	<p>Generates an XML report describing the existing business rules. See “Generating a Rule Repository Report” on page 49 in the <i>Rules Guide</i>.</p>

Command	Action
gwpc regen-soap-api	Builds the web services (SOAP) API WSDL the PolicyCenter/soap-api directory. See "Regenerating Integration Libraries and WSDL" on page 31 in the <i>Integration Guide</i> . Classes: com.guidewire.tools.wsdl.WSDLGenerator com.guidewire.util.webservices.axis.WSDLToJavaGenerator
gwpc regen-wsi-local	Regenerates WSDL for use in testing for all local web services. PolicyCenter generates the WSDL in the PolicyCenter\modules\configuration\gsrc\wsi\local directory. See "Generating WSDL On Disk" on page 54 in the <i>Integration Guide</i> .
gwpc regen-xsd	Builds the XSD files for data import. See "Constructing an XML File for Import" on page 101 in the <i>System Administration Guide</i> and "Importing Administrative Data" on page 95 in the <i>Integration Guide</i> .
gwpc studio	Runs Guidewire Studio. Class: com.guidewire.studio.main.Main
gwpc upgrade-productmodel-to-structure -DtargetPath=target_path	Upgrades a 7.0.x or 8.0.x product model structure to the current 8.0.x structure. Use this command to upgrade the product model directory specified by targetPath to the current 8.0.x product model hierarchy. Before running this command, Guidewire recommends that you make a backup copy of the specified product model directory as the command destructively modifies the target directory. The targetPath argument is case-sensitive.
gwpc verify-checksum	Verifies module checksums. Class: com.guidewire.tools.checksum.ModulesChecksumTool

Command	Action
<code>gwpc verify-resources -DmaxSPVInclusions=n</code>	<p>Checks PCF files, XML schemas, and type loaders for errors.</p> <p>This command performs PCF validation to catch errors in PCF files, such as:</p> <ul style="list-style-type: none"> • Invalid expressions • Attributes that have no meaning when another attribute is set • Editable cells within non-editable objects • Illegal use of check boxes • Invalid arguments • Other errors <p>The command parses and compiles XML schemas and reports any errors.</p> <p>For each type loader, the command verifies the types supported by the loader. The command checks if a Gosu class compiles, or if a typelist can be retrieved properly.</p> <p>Server commands that perform PCF validation, including <code>regen-dictionary</code>, <code>regen-pcfmapping</code>, and <code>verify-resources</code> can take a very long time to complete. The server performs a second-pass verification which, among other operations, verifies all possible combinations of modal sections on each PCF page. For example, a page with 12 modes that is used four times causes the server command to validate $12 \times 12 \times 12 \times 12 = 20,736$ combinations. To limit the number of shared section verifications performed by these commands, specify the optional parameter <code>maxSPVInclusions</code>. This parameter defines the depth for second pass verification that limits the number of shared sections that are included in the verification of PCF types. For instance:</p> <pre>gwpc verify-resources -DmaxSPVInclusions=1000</pre> <p>In this case, the second pass compilation of PCF files would stop after 1000 permutations of modal PCF files.</p> <p>Experiment with values for <code>maxSPVInclusions</code> between 1000 and 1000000 to achieve improved command completion times. However, be aware that limiting the validation depth means that some combinations of PCF modes and uses are not validated.</p> <p>For more information on second pass verification, see “Setting Verification Options” on page 116 in the <i>Configuration Guide</i>.</p> <p>The <code>maxSPVInclusions</code> property can only be specified as a positive integer value.</p>
<code>gwpc version</code>	Displays the product version.
<code>gwpc zip-changed-config -DoutputFile filename.zip [-DappRootDirectory PolicyCenter Home] [-DexcludeDir directory1;directory2]</code>	<p>Creates a ZIP file containing all files that are changed from the base configuration. Specify the output filename with the <code>-DoutputFile</code> parameter. You can also specify an application root directory by setting the <code>-DappRootDirectory</code> parameter. If you do not set <code>-DappRootDirectory</code>, the tool uses the directory above the <code>bin</code> directory as the root. The tool saves the output file relative to the application root. This file must not already exist.</p>

This list does not include commands specific to starting and stopping the QuickStart server. For a full list of `gwpc` commands, see “Commands Reference” on page 119.

Configuring a Database Connection

Set database connections by uncommenting and modifying the appropriate sample `<database>` element in the `database-config.xml` file, accessible from Guidewire Studio under **configuration** → **config**. In this file you supply connection and configuration parameters for your database.

This topic includes the following:

- “The <database> Element” on page 67
- “Mapping Logical Tablespaces to Physical Tablespaces” on page 68
- “Configuring Options for Individual Tables” on page 69
- “Defining Table Groups” on page 70
- “Defining the JDBC URL” on page 71
- “Specifying a Database Password” on page 71
- “Enabling SQL Server JDBC Logging” on page 72
- “Configuring PolicyCenter to Use a JNDI Data Source” on page 73
- “Creating a JNDI Data Source on JBoss” on page 74
- “Creating a JNDI Data Source on WebLogic” on page 77
- “Creating a JNDI Data Source on WebSphere” on page 79

After you have finished configuring the database connection, proceed to “Deploying PolicyCenter to the Application Server” on page 84 for instructions on deploying PolicyCenter to the application server.

The <database> Element

The <database> element in database-config.xml has the following basic structure:

```
<database name="string" env="string" dbtype="oracle|sqlserver" autoupgrade="true|false"
checker="true|false" addforeignkeys="true|false" printcommands="true|false">

  // If using database connection pool managed by PolicyCenter
  <dbcp-connection-pool jdbc-url="jdbc url" password-file="file name">
    <reset-tool-params collation="collation"
      oracle.tnsnames="Oracle TNS name" system.username="system username"
      system.password="system.password"/>
  </dbcp-connection-pool>

  // If using a JNDI data source
  <jndi-connection-pool datasource-name="JNDI data source name" />

  // Oracle only
  <oracle-settings query-rewrite="true|false" statistics-level-all="true|false"
    stored-outline-category db-resource-mgr-cancel-sql >
  <upgrade>
    <ora-db-ddl>
      <tablespaces admin="admin tablespace" index="index tablespace" op="op tablespace"
        staging="staging tablespace" typelist="typelist tablespace" lob="lob tablespace" />
    </ora-db-ddl>
  </upgrade>

  // SQL Server only
  <sqlserver-settings jdbc-trace-level="JDBC trace level" jdbc-trace-file="JDBC trace file"
    unicode-columns="true|false" />
  <upgrade>
    <mssql-db-ddl>
      <mssql-filegroups admin="admin filegroup" index="index filegroup" op="op filegroup"
        staging="staging filegroup" typelist="typelist filegroup" lob="lob filegroup" />
    </mssql-db-ddl>
  </upgrade>

  <databasesstatistics />
</database>
```

Some elements and attributes are not shown. These elements and attributes are described in other sections.

The attributes in the <database> element are:

Attribute	Description
name	A string specifying the database name.
env	A string identifying the environment in which PolicyCenter uses this connection specification. See “env Property” on page 16 in the <i>System Administration Guide</i> for information about using this attribute.
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 Data Model Updates” on page 37 in the <i>System Administration Guide</i> .
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	<p>A Boolean value specifying whether PolicyCenter runs consistency checks before it starts.</p> <p>For development environments with small data sets, you can enable consistency checks to run each time the PolicyCenter server starts. Set the checker attribute of the database block to true to enable checks on startup. By default, this option is set to false.</p> <p>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.</p> <p>This parameter is optional.</p> <p>See “Checking Database Consistency” on page 38 in the <i>System Administration Guide</i> for more information.</p>
addforeignkeys	Used only for development and testing. Do not use this attribute in production.

To improve performance of certain pages in PolicyCenter such as the **Desktop** page, add the following configuration to the <database> element in database-config.xml:

```
<databasestatistics ...>
  <tablestatistics name="pc_userroleassign">
    <histogramstatistics name="CreateTime" numbuckets="75"/>
  </tablestatistics>
</databasestatistics>
```

See also

- “Configuring the Database” on page 27
- “Configuring and Maintaining the PolicyCenter Database” on page 33 in the *System Administration Guide*

Mapping Logical Tablespaces to Physical Tablespaces

You can map the logical tablespaces required by PolicyCenter to either Oracle tablespaces or SQL Server filegroups. For Oracle, mapping to tablespaces in database-config.xml is required. For SQL Server, mapping to filegroups is optional. Create these physical tablespaces or filegroups when you set up your database. See “Configuring the Database” on page 27 for information on creating physical tablespaces or filegroups for your database.

Specifying Tablespaces for Oracle

To specify tablespaces for Oracle, use the following syntax in your database configuration in database-config.xml:

```
<database>
...
  <upgrade>
```

```

    <ora-db-ddl>
      <tablespaces admin="admin tablespace" index="index tablespace" op="op tablespace"
        staging="staging tablespace" typelist="typelist tablespace" lob="lob tablespace" />
    </ora-db-ddl>
  </upgrade>
</database>

```

To specify tablespaces for a particular table in Oracle, use the following syntax in your database configuration in `database-config.xml`:

```

<database>
  ...
  <upgrade>
    <ora-db-ddl>
      <tablespaces admin="admin tablespace" index="index tablespace" op="op tablespace"
        staging="staging tablespace" typelist="typelist tablespace" lob="lob tablespace" />
      <ora-table-ddl table-name="table name">
        <ora-table-tablespaces table-tablespace="table tablespace" lob-tablespace="LOB tablespace"
          index-tablespace="index tablespace"/>
      </ora-table-ddl>
    </ora-db-ddl>
  </upgrade>
</database>

```

Specifying Filegroups for SQL Server

To specify filegroups for SQL Server, use the following syntax in your database configuration in `database-config.xml`:

```

<database>
  ...
  <upgrade>
    <mssql-db-ddl>
      <mssql-filegroups admin="admin filegroup" index="index filegroup" op="op filegroup"
        staging="staging filegroup" typelist="typelist filegroup" lob="lob filegroup" />
    </mssql-db-ddl>
  </upgrade>
</database>

```

To specify filegroups for a particular table in SQL Server, use the following syntax in your database configuration in `database-config.xml`:

```

<database>
  ...
  <upgrade>
    <mssql-db-ddl>
      <mssql-filegroups admin="admin filegroup" index="index filegroup" op="op filegroup"
        staging="staging filegroup" typelist="typelist filegroup" lob="lob filegroup" />
      <mssql-table-ddl table-name="table name">
        <mssql-table-filegroups table-filegroup="table filegroup" lob-filegroup="LOB filegroup"
          index-filegroup="index filegroup"/>
      </mssql-table-ddl>
    </mssql-db-ddl>
  </upgrade>
</database>

```

Configuring Options for Individual Tables

You can configure DDL options for individual tables and indexes. Note however that DDL attributes of the primary key backing index (on column named with suffix ID) cannot be specified in the configuration. These attributes are defaulted by the DBMS.

This topic shows valid syntax in `database-config.xml` for configuring DDL options for each database type. All options are shown for reference. Information about the DDL options is provided in other topics, listed at the end of this topic.

Oracle

```

<database>
  ...
  <upgrade>
    <ora-db-ddl>
      <tablespaces admin="admin tablespace" index="index tablespace" op="op tablespace"
        staging="staging tablespace" typelist="typelist tablespace" lob="lob tablespace" />
      <ora-compression table-compression="ADVANCED|BASIC|NONE" index-compression="true|false">

```

```

<ora-lob type="BASIC|SECURE|SECURE_COMPRESSED" caching="true|false" />
<ora-table-ddl table-name="pc_tableName">
  <ora-index-ddl key-columns="column1,column2" index-compression="true|false"
    index-tablespace="index tablespace">
    <ora-index-hash-partitioning locality="GLOBAL|LOCAL" num-partitions="number"/>
  </ora-index-ddl>
  <ora-lob type="BASIC|SECURE|SECURE_COMPRESSED" caching="true|false" />
  <ora-table-compression table-compression="NONE|OLTP" />
  <ora-table-hash-partitioning hash-column="column name" num-partitions="number"/>
  <ora-table-tablespaces table-tablespace="table tablespace" lob-tablespace="LOB tablespace"
    index-tablespace="index tablespace"/>
</ora-table-ddl>
</ora-db-ddl>
</upgrade>
</database>

```

SQL Server

```

<database>
...
<upgrade>
  <mssql-db-ddl>
    <mssql-filegroups admin="admin filegroup" index="index filegroup" op="op filegroup"
      staging="staging filegroup" typelist="typelist filegroup" lob="lob filegroup" />
    <mssql-table-ddl table-name="table name">
      <mssql-index-ddl key-columns="column1,column2" index-compression="true|false"
        index-filegroup="index filegroup"/>
      <mssql-table-compression table-compression="NONE|PAGE|ROW"
        index-compression="NONE|PAGE|ROW" />
      <mssql-table-filegroups table-filegroup="table filegroup" lob-filegroup="LOB filegroup"
        index-filegroup="index filegroup"/>
    </mssql-table-ddl>
  </mssql-db-ddl>
</upgrade>
</database>

```

See also

- “Configuring Compression” on page 28
- “Configuring Oracle for PolicyCenter” on page 32
- “Configuring SQL Server for PolicyCenter” on page 38

Defining Table Groups

You can define zero or more table groups within the `config` element of `config.xml`. 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 `config` element in `config.xml`. The `<tablegroup>` element has an `env` attribute, a `name` attribute and a `tables` attribute. The `env` attribute specifies the environment for the table group. You can have different table groups set up for different environments. 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="pc_sometable1, pc_someTable2, pc_someTable3"/>
...
</database>

```

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

For instructions to run consistency checks from the PolicyCenter application for a particular table group, see “Consistency Checks” on page 154 in the *System Administration Guide*. For instructions to run consistency checks from a command prompt, see “System Tools Command” on page 180 in the *System Administration Guide*.

Defining the JDBC URL

The `jdbc-url` attribute of the `<dbcp-connection-pool>` element stores connection information for the database. Define a `jdbc-url` attribute 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 PolicyCenter to Use a JNDI Data Source” on page 73.

Defining JDBC URL for Oracle

Specify the JDBC URL for a standalone Oracle instance in either of the following formats:

```
<dbcp-connection-pool jdbc-url="jdbc:oracle:thin:userName/password@serverName:port/OracleSID" />

or

<dbcp-connection-pool jdbc-url="jdbc:oracle:thin:userName/password@
  (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=serverName)(PORT=port))
  (CONNECT_DATA=(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 12.1.0.1.0 Production JDBC Thin Driver, pure Java, Type IV driver in `PolicyCenter/admin/lib/ojdbc7-12.1.0.1.0.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#BEIJFHBB

Defining JDBC URL for SQL Server

Specify the JDBC URL for SQL Server in the following format:

```
<dbcp-connection-pool jdbc-url="jdbc:sqlserver://serverName[:port];
  databaseName=pc;user=pcUser;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 JDBC URL. 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.exe`. If your SQL Server instance is listening to a different port than the default 1433, specify that port number in the JDBC URL.

You can include an `applicationName` property on the JDBC URL 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 PolicyCenter server creates one by concatenating `pc`, followed by the application version, including build number. Finally, if you defined the SQL Server `ADDL_CONN_DESCR` system property, PolicyCenter appends this value to the generated `applicationName` value.

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

PolicyCenter defaults the value of the `sendStringParametersAsUnicode` property to be the correct, appropriate value in the SQL Server JDBC URL connection. PolicyCenter does not override an existing value. If you set `sendStringParametersAsUnicode` in `database-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 `database-config.xml` file, set the password in the JDBC URL. If you do not want to expose this password in `database-config.xml`, Guidewire provides the following alternatives:

- “Using a Password File” on page 72
- “Using the Database Authentication Plugin” on page 72
- “Using a JNDI Data Source” on page 72

Using a Password File

You can place the password in an external file and reference this file from the `database-config.xml` file.

To use a password file

1. Add the `password-file` attribute to the `<dbcp-connection-pool>` element within the `<database>` element.
2. Set the value of `password-file` to the absolute path of the password file.
3. Replace the password value in the `jdbc-url` connection specification with a `${password}` placeholder.

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

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

Oracle:

```
<database name="BillingCenterDatabase" driver="dbcp" dbtype="oracle" autoupgrade="true">
  <dbcp-connection-pool jdbc-url="jdbc:oracle:thin:USER/${password}@ORACLEDB:PORT:INSTANCE"
    password-file="c:\secure\password.txt" />
</database>
```

SQL Server:

```
<database name="PolicyCenterDatabase" driver="dbcp"
  dbtype="sqlserver" autoupgrade="true">
  <dbcp-connection-pool jdbc-url="jdbc:sqlserver://HOSTNAME:1433;databaseName=pc;user=pcUser;
    password=${password}" password-file="c:\secure\password.txt" />
</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 189 in the *Integration Guide*.

Using a JNDI Data Source

You can configure PolicyCenter to use a JNDI data source on a JBoss, 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 PolicyCenter to Use a JNDI Data Source” on page 73 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 PolicyCenter by specifying the `jdbc-trace-file` and `jdbc-trace-level` attributes on the `<sqlserver-settings>` element:

```
<database ...>
  <sqlserver-settings jdbc-trace-file="file name" jdbc-trace-level="trace level" />
</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](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 PolicyCenter 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 `database-config.xml` or previous use of this page, then the logging level resets to the new level. PolicyCenter 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 PolicyCenter only works when using the internal connection pool, not when using an external JNDI data source connection pool.

Configuring PolicyCenter to Use a JNDI Data Source

PolicyCenter can use a Java Naming and Directory Interface (JNDI) data source managed by a JBoss, Tomcat, 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 `database-config.xml` file.

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

By default, the database connection in `database-config.xml` defines a Java Database Connectivity (JDBC) specification for connecting to the database. To specify a JNDI data source connection, replace the `<dbcp-connection-pool>` element on your `<database>` element with a `<jndi-connection-pool>` element.

To configure PolicyCenter to use a JNDI data source

1. Open `database-config.xml` from the Guidewire Studio Project window under **configuration** → **config**.
2. Remove the `<dbcp-connection-pool>` element.
3. Add a `<jndi-connection-pool>` element and specify the JNDI name you assign to the data source as a `datasource-name` attribute.

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

```
<database name="PolicyCenterDatabase" dbtype="oracle|sqlserver"
  autoupgrade="false">
  <jndi-connection-pool datasource-name="jdbc/pcDataSource" />
  ...
</database>
```

This example is for a direct JNDI lookup. If you want to use an indirect JNDI lookup, use the format `java:comp/env/jdbc/DataSourceName` for the `datasource-name` value, replacing `DataSourceName` with the name you assigned to the data source.

4. Close and save the `database-config.xml` file.
5. Rebuild and install the PolicyCenter application EAR or WAR file. See “Deploying PolicyCenter to the Application Server” on page 84 for instructions.

Before deploying PolicyCenter 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 JBoss” on page 74

- “Creating a JNDI Data Source on Tomcat” on page 76
- “Creating a JNDI Data Source on WebLogic” on page 77
- “Creating a JNDI Data Source on WebSphere” on page 79

During startup, PolicyCenter 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/pcDataSource'...
```

Creating a JNDI Data Source on JBoss

This topic describes how to create a JNDI data source on JBoss. To configure PolicyCenter to use the data source, see “Configuring PolicyCenter to Use a JNDI Data Source” on page 73.

This topic includes the following:

- “Creating an Oracle JNDI Data Source on JBoss” on page 74
- “Creating a SQL Server JNDI Data Source on JBoss” on page 75

Creating an Oracle JNDI Data Source on JBoss

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

To create an Oracle JNDI data source on JBoss

1. Start the JBoss application server.
2. Log in to the JBoss Admin Console.
3. Click **Runtime**.
4. Click **Manage Deployments** for the domain hosting PolicyCenter. For standalone installations, no domain is shown.
5. Click **Add**.
6. Click **Browse**.
7. Select the `ojdbc7-12.1.0.1.0-prod.jar` file from the PolicyCenter `admin/lib` directory and click **Next**.
8. Click **Save**.
9. For managed domain operating mode, assign the JAR.
 - a. Click **Assign**.
 - b. Select `main-server-group` and ensure `Enable ojdbc7-12.1.0.1.0-prod.jar` is selected.
 - c. Click **Save**.
10. Click **Configuration**.
11. Select the profile to configure.
12. Click **Subsystems** → **Connector** → **Datasources**.
13. Click **Add**.
14. Enter a **Name** for the datasource.
15. Enter a **JNDI Name**. The JNDI name must use the pattern `java:jboss/datasources/name`. The JNDI name must match the value of the `datasource-name` attribute of the `<jndi-connection-pool>` element in the `database-config.xml` file.
16. Click **Next**.

17. Select the Oracle JDBC driver and click **Next**.
18. Enter the **Connection URL**, **Username**, and **Password** and click **Done**.
19. Click **Validation** and click **Edit**.
20. Select **Background Validation**.
21. Set **Validation Millis** to 5000.
22. Set **Check Valid SQL** to `select 1 from dual`.
23. Select the datasource and click **Enable**.
24. Select **Connection** and click **Test Connection**. If you do not receive a message that the JDBC connection was successful, check your connection settings.

Creating a SQL Server JNDI Data Source on JBoss

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

To create a SQL Server JNDI data source on JBoss

1. Start the JBoss application server.
2. Log in to the JBoss Admin Console.
3. Click **Runtime**.
4. Click **Manage Deployments** for the domain hosting PolicyCenter. For standalone installations, no domain is shown.
5. Click **Add**.
6. Click **Browse**.
7. Select the `sqljdbc4-4.0.2206.100.jar` file from the PolicyCenter `admin/lib` directory and click **Next**.
8. Click **Save**.
9. For managed domain operating mode, assign the JAR.
 - a. Click **Assign**.
 - b. Select `main-server-group` and ensure **Enable sqljdbc4-4.0.2206.100.jar** is selected.
 - c. Click **Save**.
10. Click **Configuration**.
11. Select the profile to configure.
12. Click **Subsystems** → **Connector** → **Datasources**.
13. Click **Add**.
14. Enter a **Name** for the datasource.
15. Enter a **JNDI Name**. The JNDI name must use the pattern `java:jboss/datasources/name`. The JNDI name must match the value of the `datasource-name` attribute of the `<jndi-connection-pool>` element in the `database-config.xml` file.
16. Click **Next**.
17. Select the SQL Server JDBC driver and click **Next**.
18. Enter the **Connection URL**, **Username**, and **Password** and click **Done**.
19. Click **Validation** and click **Edit**.

20. Select **Background Validation**.
21. Set **Validation Millis** to 5000.
22. Set **Check Valid SQL** to `select 1 where 1 = 0`.
23. Select the datasource and click **Enable**.
24. Select **Connection** and click **Test Connection**. If you do not receive a message that the JDBC connection was successful, check your connection settings.

Creating a JNDI Data Source on Tomcat

This topic describes how to create a JNDI data source on Tomcat. To configure PolicyCenter to use the data source, see “Configuring PolicyCenter to Use a JNDI Data Source” on page 73.

This topic includes the following:

- “Creating an Oracle JNDI Data Source on Tomcat” on page 76
- “Creating a SQL Server JNDI Data Source on Tomcat” on page 76

See also

- <http://commons.apache.org/proper/commons-dbcp/configuration.html> for more info about configuring the data source.

Creating an Oracle JNDI Data Source on Tomcat

1. Copy the `ojdbc7-12.1.0.1.0-prod.jar` file from the `PolicyCenter/admin/lib` directory to the `lib` directory within the Tomcat home. The `ojdbc7-12.1.0.1.0-prod.jar` file contains the APIs for connecting to the PolicyCenter database.
2. Add a Resource entry to the `context.xml` file in the `conf` directory of the Tomcat instance. If the `context.xml` file does not exist, create it. The Resource element must be a child of the top-level Context element. The resource name must match the `datasource-name` attribute of the `jndi-connection-pool` element in `database-config.xml`. For example:

```
<Resource name="jdbc/pcDataSource" auth="Container" type="javax.sql.DataSource"
  driverClassName="oracle.jdbc.OracleDriver"
  url="jdbc:oracle:thin:database user/database password@database server name:port/OracleSID"
  username="database user" password="database password" maxActive="20" maxIdle="10" maxWait="-1"/>
```

3. Add a resource-ref entry to the `web.xml` file in the `conf` directory of the Tomcat instance. For example:

```
<resource-ref>
  <description>Oracle Datasource</description>
  <res-ref-name>jdbc/Datasource name</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

4. Restart the Tomcat instance.

Creating a SQL Server JNDI Data Source on Tomcat

1. Copy the `sqljdbc4-4.0.2206.100.jar` file from the `PolicyCenter/admin/lib` directory to the `lib` directory within the Tomcat home. The `sqljdbc4-4.0.2206.100.jar` file contains the APIs for connecting to the PolicyCenter database.
2. Add a Resource entry to the `context.xml` file in the `conf` directory of the Tomcat instance. If the `context.xml` file does not exist, create it. The Resource element must be a child of the top-level Context element. The resource name attribute must match the `datasource-name` attribute of the `jndi-connection-pool` element in `database-config.xml`. For example:

```
<Resource name="jdbc/pcDataSource" auth="Container" type="javax.sql.DataSource"
  driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
  url="jdbc:sqlserver://database server name:database server port;selectMethod=direct;
```

```

        databaseName=database name;sendStringParametersAsUnicode=false;user=database user;
        password=database password"
    username="database user" password="database password" maxActive="20" maxIdle="10" maxWait="-1"/>

```

3. Add a resource-ref entry to the web.xml file in the conf directory of the Tomcat instance. For example:

```

<resource-ref>
  <description>SQL Server Datasource</description>
  <res-ref-name>jdbc/Datasource name</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>

```

4. Restart the Tomcat instance.

Creating a JNDI Data Source on WebLogic

This topic describes how to create a JNDI data source on WebLogic. To configure PolicyCenter to use the data source, see “Configuring PolicyCenter to Use a JNDI Data Source” on page 73.

This topic includes the following:

- “Creating an Oracle JNDI Data Source on WebLogic” on page 77
- “Creating a SQL Server JNDI Data Source on WebLogic” on page 78

Creating an Oracle JNDI Data Source on WebLogic

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

To copy the ojdbc7-12.1.0.1.0-prod.jar file to WebLogic

1. Copy the ojdbc7-12.1.0.1.0-prod.jar file from the PolicyCenter/admin/lib directory to the server/lib directory within the WebLogic home. The ojdbc7-12.1.0.1.0-prod.jar file contains the APIs for connecting to the PolicyCenter database.
2. Delete the ojdbc6.jar file from the WebLogic server/lib directory.
3. Add the ojdbc7-12.1.0.1.0-prod.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 ojdbc7-12.1.0.1.0-prod.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 ojdbc7-12.1.0.1.0-prod.jar file to the `PRE_CLASSPATH` environment variable. The WebLogic server must be restarted for this change to take effect.
4. Restart WebLogic.

To create the data source

1. Open the WebLogic Server Administration Console.
2. Choose **Service** → **Data Source**.
3. Click **Lock & Edit**.
4. Click **New** to create a Generic Data Source.
5. Enter a name and JNDI name for the data source. The JNDI name must match the value of the `datasource-name` attribute of the `<jndi-connection-pool>` element in the `database-config.xml` file. The JNDI name typically begins with `jdbc/`.
6. Select **Oracle** as the **Database Type**.
7. Select **Oracle's Driver (Thin)** for **Service Connections**.

8. Select **Versions:9.0.1 and later** for **Database Driver** and click **Next**.
9. Fill in connection properties for your environment and click **Next**. The WebLogic Server Administration Console displays the **Test Database Connection** page.
10. Leave **Driver Class Name** as `oracle.jdbc.OracleDriver`.
11. Click **Test Configuration**. If you configured the connection properly and the database is running, WebLogic displays the message “Connection test succeeded.”
12. Click **Next**.
13. Select the servers that use the data source.
14. Click **Finish**. The WebLogic Server Administration Console returns you to the **Summary of JDBC Data Sources** page.
15. 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 `sqljdbc4-4.0.2206.100.jar` file to WebLogic

1. Copy the `sqljdbc4-4.0.2206.100.jar` file from the `PolicyCenter/admin/lib` directory to the `server/lib` directory within the WebLogic home. The `sqljdbc4-4.0.2206.100.jar` file contains the APIs for connecting to the PolicyCenter database.
2. Add the `sqljdbc4-4.0.2206.100.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 `sqljdbc4-4.0.2206.100.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 `sqljdbc4-4.0.2206.100.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 **Service** → **Data Source**.
3. Click **New** to create a Generic Data Source.
4. Enter a **Name** and **JNDI Name** for the data source. The JNDI name must match the value of the `datasource-name` attribute of the `<jndi-connection-pool>` element in the `database-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 PolicyCenter to use the data source, see “Configuring PolicyCenter to Use a JNDI Data Source” on page 73.

This topic includes the following:

- “Creating an Oracle JNDI Data Source on WebSphere” on page 79
- “Creating a SQL Server JNDI Data Source on WebSphere” on page 81

Creating an Oracle JNDI Data Source on WebSphere

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

To copy the `ojdbc7-12.1.0.1.0-prod.jar` file to WebSphere

Before you begin, copy the `ojdbc7-12.1.0.1.0-prod.jar` file from `PolicyCenter/admin/lib` to the WebSphere `WAS_HOME/lib/ext` directory. The `ojdbc7-12.1.0.1.0-prod.jar` file contains the APIs for connecting to the PolicyCenter 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 **Oracle** for the **Database type**.
6. Select **Oracle JDBC Driver** for the **Provider type**.
7. Select **Connection pool data source** for the **Implementation type**.
8. Supply a new **Name** for the JDBC provider, for example `pcOracle`.
9. Enter a **Description** for the JDBC provider if you want.
10. Click **Next**.
11. Specify the directory location of `ojdbc7-12.1.0.1.0-prod.jar`. Set the value to the `WAS_HOME/lib/ext` path where you copied `ojdbc7-12.1.0.1.0-prod.jar` earlier.
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. If you do not yet have a J2C (Java 2 Connector) authentication alias, create a new one.
 - a. Click **Security** → **Global security**.
 - b. Under **Authentication** click **Java Authentication and Authorization** → **J2C authentication data**.
 - c. Click **New**.
 - d. Enter the following.

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

- e. Click **OK**.
 - f. Click **Save** to save apply your changes to the master configuration.
 - g. Start the procedure to create the data source again, beginning with step 3.
3. Choose **Resources** → **JDBC** → **JDBC Providers**.
4. Set the **Scope** to **Cell**.
5. Select the JDBC provider that you defined for Oracle. WebSphere displays the **Configuration** tab.
6. Select **Data Sources** in the **Additional Properties** section. WebSphere displays the **Data sources** page.
7. Click **New** to create a new data source.
8. Enter a **JNDI name** for the data source. The JNDI name must match the value of the `datasource-name` attribute of the `<jndi-connection-pool>` element in the `database-config.xml` file. See “Configuring PolicyCenter to Use a JNDI Data Source” on page 73.
9. Click **Next**.
10. Set the **URL** value using the `jdbc:oracle:thin:@hostname:port:ORACLE_SID` format.
11. Select **Oracle11g data store helper** for the **Data store helper class name**.
12. Click **Next**.
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 PolicyCenter 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 `sqljdbc4-4.0.2206.100.jar` file to WebSphere

Before you begin, copy the `sqljdbc4-4.0.2206.100.jar` file from `PolicyCenter/admin/lib` to the WebSphere `lib/ext` directory. The `sqljdbc4-4.0.2206.100.jar` file contains the APIs for connecting to the PolicyCenter 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 `pcSQLServer`.
9. Enter a **Description** for the JDBC provider if you want.
10. Click **Next**.
11. Specify the directory location of `sqljdbc4-4.0.2206.100.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 `sqljdbc4-4.0.2206.100.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}/sqljdbc4-4.0.2206.100.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. 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 <code>pcAlias</code> .
User ID	A string specifying the user name.
Password	A string specifying the password.

d. Click **OK**.

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

f. Start the procedure to create the data source again, beginning with step 3.

3. Choose **Resources** → **JDBC** → **JDBC Providers**.

4. Set the **Scope** to **Cell**.

5. Select the JDBC provider that you created for SQL Server.

WebSphere displays the **Configuration** tab.

6. Select **Data Sources** in the **Additional Properties** section.

WebSphere displays the **Data sources** page.

7. Click **New** to create a new data source.

Enter a **JNDI name** for the data source, such as `jdbc/pcDataSource`. The JNDI name must match the value of the `datasource-name` attribute of the `<jndi-connection-pool>` element in the `database-config.xml` file. See “Configuring PolicyCenter to Use a JNDI Data Source” on page 73.

8. Click **Next**.

9. Enter the **Database name**.

10. If using a different port to connect to the database than the default of 1433, change the **Port number** value.
11. Enter the **Server name** for the server hosting SQL Server.
12. Uncheck the box for **Use this data source in container managed persistence (CMP)**.
13. Click **Next**.
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 PolicyCenter data source.
3. Click **Test Connection** to verify that the connection works.

Deploying PolicyCenter to the Application Server

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

These instructions are for creating a production PolicyCenter environment. For instructions to create a development PolicyCenter environment, see “Installing a PolicyCenter Development Environment” on page 47.

This topic includes the following:

- “Installing a JBoss Production Environment” on page 84
- “Installing a Tomcat Production Environment” on page 84
- “Installing a WebLogic Production Environment” on page 85
- “Installing a WebSphere Production Environment” on page 87

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 PolicyCenter to JBoss, add any additional servlets, then generate and deploy the WAR file. For instructions to deploy the PolicyCenter WAR file, refer to the JBoss Enterprise Application Platform *Administration and Configuration Guide*.

To add servlet definitions

1. Start Studio by running the following command from the `PolicyCenter/bin` directory:

```
gwpc studio
```

2. Expand **configuration** → **deploy** → **WEB-INF** and open `web.xml`.
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 your changes.

Review the post-installation tasks in “Additional PolicyCenter Setup Tasks” on page 89. Then proceed to “Starting PolicyCenter on JBoss” on page 116.

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
```

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. In Studio, expand **configuration** → **deploy** → **WEB-INF** and open `web.xml`.
2. Add a servlet-mapping definition for each servlet that you add. See the defined servlet mappings for an example.
3. Save `web.xml`.

To deploy to Tomcat

1. Open a command window.
2. Navigate to `PolicyCenter\bin`.
3. Run one of the following commands:

```
gwpc build-tomcat-war-dbc  
gwpc build-tomcat-war-jndi
```

Both commands generate the PolicyCenter WAR file in the PolicyCenter `dist/war` directory. Run `gwpc build-tomcat-war-dbc` to build the EAR file with JDBC drivers for use with Tomcat. Run `gwpc build-tomcat-war-dbc` if you are going to have PolicyCenter manage the database connection pool. Run `gwpc build-tomcat-war-jndi` to build the EAR file without JDBC drivers. Use `gwpc build-tomcat-war-jndi` only if you are going to use a JNDI database connection managed by Tomcat. See “Using a JNDI Data Source” on page 72.

4. Deploy the package to Tomcat by copying the `pc.war` file to the `webapps` directory in your Tomcat server. When Tomcat starts up, it automatically recognizes PolicyCenter and unpacks the `pc.war` into a directory structure within `Tomcat\webapps`. For this example, Tomcat creates a `Tomcat\webapps\pc` directory. Each time you deploy a new copy of a `pc.war` file, delete the pre-existing `pc` directory structure created by the old `pc.war` file.

Review the post-installation tasks in “Additional PolicyCenter Setup Tasks” on page 89. Then proceed to “Starting PolicyCenter on Tomcat on Windows” on page 116.

Installing a WebLogic Production Environment

For WebLogic, PolicyCenter 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 85
- “Generating the PolicyCenter EAR file for WebLogic” on page 86
- “Installing the EAR file on WebLogic” on page 86
- “Automatic Versus Manual Startup on WebLogic” on page 87

Adding Servlet Definitions for WebLogic

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

To add servlet definitions

1. Start Studio by running the following command from the `PolicyCenter/bin` directory:

```
gwpc studio
```
2. Expand **configuration** → **deploy** → **WEB-INF** and open `web.xml`.
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.

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>
```

Generating the PolicyCenter EAR file for WebLogic

This procedure builds the PolicyCenter EAR file.

To generate the PolicyCenter EAR file

1. Open a command prompt.
2. Navigate to the PolicyCenter `bin` directory.
3. Run one of the following commands:

```
gwpc build-weblogic-ear-dbc  
gwpc build-weblogic-ear-jndi
```

Both commands generate the PolicyCenter EAR file in the PolicyCenter `dist/ear` directory. Run `gwpc build-weblogic-ear-dbc` to build the EAR file with JDBC drivers for use with WebLogic. Run `gwpc build-weblogic-ear-dbc` if you are going to have PolicyCenter manage the database connection pool. Run `gwpc build-weblogic-ear-jndi` to build the EAR file without JDBC drivers. Use `gwpc build-weblogic-ear-jndi` only if you are going to use a JNDI database connection managed by WebLogic. See “Using a JNDI Data Source” on page 72.

Installing the EAR file on WebLogic

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

To install the PolicyCenter 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 PolicyCenter `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 starts PolicyCenter 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 PolicyCenter manually. See “Starting PolicyCenter on WebLogic” on page 117.

Installing a WebSphere Production Environment

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

Guidewire only supports externally managed data sources using JDBC drivers shipped with PolicyCenter 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 the *Guidewire Platform Support Matrix* for details.

To add a welcome-file-list

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

1. Start Studio by running the following command from the `PolicyCenter/bin` directory:

```
gwpc studio
```

2. Expand **configuration** → **deploy** → **WEB-INF** and open `web.xml`.

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 PolicyCenter EAR File for WebSphere

1. Open a command window.
2. Navigate to the PolicyCenter `bin` directory.
3. Run one of the following commands:

```
gwpc build-websphere-ear-dbc
gwpc build-websphere-ear-jndi
```

These commands build the EAR file and place it in `PolicyCenter\dist\ear`. The commands package the version of `web.xml` from the `PolicyCenter\modules\configuration\deploy\WEB-INF` directory.

Run `gwpc build-websphere-ear-dbc` to build the EAR file with JDBC drivers for use with WebSphere. Use `gwpc build-websphere-ear-dbc` if you are going to have PolicyCenter manage the database connection pool. Run `gwpc build-websphere-ear-jndi` to build the EAR file without JDBC drivers. Use `gwpc build-websphere-ear-jndi` only if you are going to use a JNDI database connection managed by WebSphere. See “Using a JNDI Data Source” on page 72.

To install the PolicyCenter EAR file on WebSphere

1. If WebSphere is not already running, start the application server.
2. Open the WebSphere Administrative Console.
3. Click **Applications** → **New Application**.
4. Click **New Enterprise Application**.
5. Click **Browse** and select the PolicyCenter EAR file in the PolicyCenter/dist/ear directory.
6. Click **Next**.
7. Select **Fast Path**.
8. Click **Next**.
9. Accept the default installation options.
10. Click **Next**.
11. On the **Map modules to servers** page, verify that your targeted server or cluster is selected.
12. Click **Next**.
13. On the **Metadata for modules** page, click **Next**.
14. On the **Summary** page, review your selections for accuracy. Click **Previous** to change any settings.
15. Click **Finish**.
16. Click **Save** to apply the changes to the master configuration.
17. Click the **pc** application.
18. Under **Detail properties**, click **Class loading and update detection**.
19. Under **Class loader order**, verify **Classes loaded with local class loader first (parent last)** is selected.
20. Under **WAR class loader policy**, verify **Single class loader for application** is selected.
21. Click **OK**.
22. Click **Save** to apply the changes to the master configuration.
23. If using JNDI, remove the JDBC JAR files (such as `ojdbc<version>.jar` or `sqljdbc<version>.jar`) from the exploded deployment in `<WAS_Profile>/installedApps/DefaultCell/pc.ear/pc.war/WEB-INF/lib`.

After you have finished the procedure to install the PolicyCenter EAR file on WebSphere, review post-installation tasks in “Additional PolicyCenter Setup Tasks” on page 89. Then, proceed to “Starting PolicyCenter on WebSphere” on page 117.

Additional PolicyCenter Setup Tasks

This topic describes additional setup tasks that you may want to perform after you install your PolicyCenter development or production environment and deploy PolicyCenter to your application server.

This topic includes:

- “Free-text Search Setup” on page 89
- “Changing the Superuser Password” on page 102
- “Generating Java and SOAP API Libraries” on page 102
- “Enabling Integration between ClaimCenter and PolicyCenter” on page 102
- “Enabling Integration between BillingCenter and PolicyCenter” on page 107
- “Enabling Archiving or Disabling Archiving Work Queues” on page 111
- “Installing Rating Management” on page 112
- “Installing Reinsurance Management or Disabling Work Queue” on page 112
- “Upgrading Product Model of Old LOB Extension Packs” on page 113
- “Configuring Single Sign-on Authentication” on page 113
- “Starting PolicyCenter on the Application Server” on page 116
- “Connecting to PolicyCenter with a Web Client” on page 117
- “Configuring Windows Accessibility for Firefox” on page 118
- “Additional Installation Information” on page 118

Free-text Search Setup

Guidewire supports deployment of the full-text search engine that free-text search depends on in JBoss, Tomcat, WebLogic, and WebSphere application servers. The full-text search engine is a special distribution of Apache Solr, tailored by Guidewire to work with free-text search. This special distribution is called the *Guidewire Solr Extension*.

This topic includes:

- “Free-text Search Setup Overview” on page 90
- “Setting Up Free-text Search for Embedded Operation” on page 92
- “Setting Up Free-text Search for JBoss” on page 93
- “Setting Up Free-text Search for Tomcat” on page 95
- “Setting Up Free-text Search for WebLogic” on page 96
- “Setting Up Free-text Search for WebSphere” on page 97
- “Setting Up the Free-text Batch Load Command” on page 100

See also

- “Free-text Search Configuration” on page 357 in the *Configuration Guide*.
- For specific versions of application servers Guidewire that supports, see the Guidewire Platform Support Matrix on the Guidewire Resource Portal, at <https://guidewire.custhelp.com/app/resources/products/platform>.

Free-text Search Setup Overview

PolicyCenter free-text search depends on a full-text search engine, the Guidewire Solr Extension. Guidewire supports running the Guidewire Solr Extension in a JBoss, Tomcat, WebLogic, or WebSphere application server in a production environment. In a development environment, Guidewire supports running the Guidewire Solr Extension in the bundled Quickstart server, if you configure free-text search for embedded operation.

See also

- “Free-text Search System Architecture” on page 358 in the *Configuration Guide*

Operating Options for Production and Development

In a production environment, you must configure the free-text search for *external* operation. With external operation in a production environment, the Guidewire Solr Extension must run in a different instance of the application server than the instance that runs your PolicyCenter application.

In a development environment, you can configure free-text search for external or *embedded* operation. With external operation in a development environment, the Guidewire Solr Extension runs in a different instance of the application server than the instance that runs your PolicyCenter application. With embedded operation, the Guidewire Solr Extension runs automatically as part of the PolicyCenter application in the application server instance that runs PolicyCenter, not as a separate application.

Note: You can configure the Guidewire Solr Extension for embedded or external operation in development environments installed on a Tomcat application server. In development environments installed on the bundled QuickStart application server, you can configure the Guidewire Solr Extension for embedded operation only.

Simplified Free-text Search Setup With Embedded Operation

In a development environment, set up the free-text search for embedded operation to simplify your setup procedure. With embedded operation you can avoid the following set-up tasks that external operation requires:

- Setting up a separate application server instance for the Guidewire Solr Extension
- Setting up the free-text batch load command

With embedded operation, your only set-up task involves changes to the `solrserver-config.xml` file. This configuration file tells the free-text feature in PolicyCenter how to operate and interact with the Guidewire Solr Extension.

See also

- “Setting Up Free-text Search for Embedded Operation” on page 92

The Guidewire Solr Extension

Guidewire provides a special distribution of the Apache Solr full-text search engine, the Guidewire Solr Extension. Guidewire provides the Guidewire Solr Extension in a ZIP file located in your installation of PolicyCenter at:

```
PolicyCenter/solr/pc-gwsolr.zip
```

The Guidewire Solr Extension runs as a web application in a JBoss, Tomcat, WebLogic, or WebSphere application server. If you configure the Guidewire Solr Extension for embedded operation, it operates as part of PolicyCenter, not as a separate web application.

WARNING Do not use any distribution of the Apache Solr full-text search engine other than the one that Guidewire provides as the Guidewire Solr Extension.

The Guidewire Solr Home Directory

In the instructions that follow, you create an installation directory for the Guidewire Solr Extension. The installation directory is known as the *Guidewire Solr home directory*. Guidewire requires that you create the Guidewire Solr home directory on the host where you installed the separate application server instance for it.

Note: Do not create a Guidewire Solr home directory if you configure free-text search for embedded operation.

The default parameter settings and configuration files for free-text search assume the following directories for the Guidewire Solr home directory:

- **On Unix** – /opt/gwsolr
- **On Windows** – C:\opt\gwsolr

The instructions that follow use /opt/gwsolr for the Guidewire Solr home directory. If you use a different directory than one of those listed above, you must perform additional configuration steps.

The Free-text Batch Load Command

The free-text search feature provides a command-line utility to extract policy data from the PolicyCenter relational database and load the extracted data into the Guidewire Solr Extension. In the instructions that follow, you unpack the files from the Guidewire Solr WAR file, which the free-text batch load command depends on.

Note: Do not set up and configure the free-text batch load command if you configure free-text search for embedded operation. Instead, use the Solr Data Import batch process.

The free-text batch load command also depends on a Unix-compatible sort binary that performs character-value sorting. The batch load command builds intermediate index documents from policy data in PolicyCenter. The batch load command sorts the intermediate documents during its process of collating and compiling the final index documents that it loads into the Guidewire Solr Extension. Guidewire supports cygwin on Windows.

In the instructions that follow, you first perform a procedure to set up free-text search for JBoss, Tomcat, WebLogic, or WebSphere. Then, regardless of application server, you perform a separate, follow-on procedure to set up the free-text batch load command.

If you set up a development environment for PolicyCenter, you can avoid setting up the free-text batch load command. Instead, use the **Free-text Search** page from the **Server Tools** tab to perform the same function. Also, the **Free-text Search** page has a function to confirm that the policy data in the Guidewire Solr Extension matches the policy data in the application database. Use the consistency checking function after you run the batch load command to verify changes to the command that you are developing and testing.

See also

- “Setting Up the Free-text Batch Load Command” on page 100
- “Configuring the Free-text Batch Load Command” on page 369 in the *Configuration Guide*
- For information on the **Free-text Search** page, see “Free-text Search” on page 170 in the *System Administration Guide*

Setting Up Free-text Search for Embedded Operation

Set up free-text search for embedded operation for preliminary testing with small amounts of test data. For final testing with larger volumes of data and to prepare for production, set up free-text search for external operation. You can set up free-text search for embedded operation in development environments only.

Setting up free-text search for embedded operation on QuickStart

You can set up free-text search in a development environment on QuickStart for embedded operation only.

1. In the **Project** window in Studio, navigate to **configuration** → **config** → **solr**, and then open `solrserver-config.xml`.

2. Verify that the file contains a `<solrserver>` element that matches the following:

```
<solrserver name="embedded" type="embedded">
  <param name="provision" value="true"/>
  <param name="solrroot" value="/opt/gwsolr"/>
</solrserver>
```

3. In the `<document>` element, change the `servername` attribute to `embedded`.

```
<document name="policy" archive="false" servername="embedded"/>
```

4. Save your changes to the `solrserver-config.xml` file.

5. Open a command prompt to `PolicyCenter/bin` and run the following command:

```
gwpc solr
```

6. Proceed to “Enabling Free-text Search in PolicyCenter” on page 362 in the *Configuration Guide*.

Setting up free-text search for embedded operation on Tomcat

You can setup free-text search in a development environment on Tomcat for embedded or external operation. For embedded operation, you must include the `gwsolrzip` parameter in the `solrserver` element to specify the absolute path to the `pc-gwsolr.zip` file in your PolicyCenter home directory.

1. In the **Project** window in Studio, navigate to **configuration** → **config** → **solr**, and then open `solrserver-config.xml`.

2. Verify that the file contains a `<solrserver>` element that matches the following:

```
<solrserver name="embedded" type="embedded">
  <param name="provision" value="true"/>
  <param name="solrroot" value="/opt/gwsolr"
  <param name="gwsolrzip" value="/dev/PolicyCenter/solr/pc-gwsolr.zip">
</solrserver>
```

3. In the `<document>` element, change the `servername` attribute to `embedded`.

```
<document name="policy" archive="false" servername="embedded"/>
```

4. Save your changes to the `solrserver-config.xml` file.

5. Open a command prompt to `PolicyCenter/bin` and run the following command:

```
gwpc solr
```

6. Proceed to “Enabling Free-text Search in PolicyCenter” on page 362 in the *Configuration Guide*.

See also

- To setup free-text search for external operation on Tomcat, see “Setting Up Free-text Search for Tomcat” on page 95.

Setting Up Free-text Search for JBoss

PolicyCenter free-text search requires that you install an instance of JBoss separate from the instance that runs your PolicyCenter application.

To set up free-text search for JBoss

1. On the host where the JBoss instance for free-text search resides, create an installation directory to use as the Guidewire Solr home directory. For example:
 - **On Unix** – /opt/gwsolr
 - **On Windows** – C:\opt\gwsolr

This guide uses /opt/gwsolr as the directory name.

2. Create the environment variable GWSOLR_HOME for the directory that you created in step 1.
3. Create a new opt/gwsolr/pc directory and extract to it the file PolicyCenter/solr/pc-gwsolr.zip from the host where your PolicyCenter application resides.
4. Copy the file /opt/gwsolr/pc/pc-gwsolr.war to the directory *JBoss_HOME*/standalone/deployments.
5. Edit the following file:

JBoss_HOME/standalone/configuration/standalone.xml

Change the port property for the WebServer service from 8080 to 8983, as the following example shows.

```
<socket-binding-group name="standard-sockets" default-interface="public"
  port-offset="${jboss.socket.binding.port-offset:0}">
  ...
  <socket-binding name="http" port="8983"/>
  ...
</socket-binding-group>
```

The standard Solr port is 8983.

6. Check the web application deployment descriptor in autodeploy/pc-gwsolr/WEB-INF/web.xml for the definition of the solr.home environment variable:

```
<env-entry>
  <env-entry-name>solr/home</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>/opt/gwsolr/pc/solr</env-entry-value>
</env-entry>
```

If you installed the ZIP file in a directory other than /opt/gwsolr, you must now change the path in the <env-entry-value> XML element to match the actual install directory path.

JBoss uses Java Logging (JUL) and the Guidewire Solr Extension uses Simple Logging Facade for Java (SLF4J). To configure SLF4J to bind to JUL, perform step 7 through step 9.

7. Create a lib folder in *GWSOLR_HOME*\pc and place the following logging JAR files there:

- jcl-over-slf4j-1.7.5.jar – Jakarta commons logging over SLF4J
- log4j-over-slf4j-1.7.5.jar – Log4j logging over SLF4J
- slf4j-api-1.7.5.jar – SLF4J API
- slf4j-ext-1.7.5.jar – SLF4J Extensions
- slf4j-jdk14-1.7.5.jar – SLF4J Java logging binding

Logging JAR files are provided in the logging_jars folder at the root of the pc-gwsolr.zip file.

IMPORTANT Install only the logging JARs specified above.

8. Add a deployment descriptor to the solr.war file.

a. In the Project window in Studio, navigate to configuration → config → solr.

b. Create a file named jboss-deployment-structure.xml, with the following content.

```
<?xml version='1.0' encoding='UTF-8'?>
<jboss-deployment-structure xmlns="urn:jboss:deployment-structure:1.1">
  <deployment>
    <dependencies>
      <module name="org.slf4j"/>
      <module name="org.slf4j.ext"/>
      <module name="org.slf4j.slf4j-jdk14"/>
      <module name="org.slf4j.jcl-over-slf4j"/>
      <module name="org.slf4j.log4j-over-slf4j"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

9. Create two modules in your JBoss configuration.

a. In JBOSS_HOME\modules, create file org\slf4j\log4j-over-slf4j\main\module.xml, with the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.slf4j.log4j-over-slf4j">
  <resources>
    <resource-root path="log4j-over-slf4j-1.7.5.jar"/>
  </resources>
  <dependencies>
    <module name="org.slf4j"/>
    <module name="org.slf4j.slf4j-jdk14"/>
  </dependencies>
</module>
```

b. Copy logging JAR log4j-over-slf4j-1.7.5.jar to the same location.

c. In JBOSS_HOME\modules, create file org\slf4j\slf4j-jdk14\main\module.xml, with the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.slf4j.slf4j-jdk14">
  <resources>
    <resource-root path="slf4j-jdk14-1.7.5.jar"/>
  </resources>
  <dependencies>
    <module name="org.slf4j"/>
  </dependencies>
</module>
```

d. Copy logging JAR slf4j-jdk14-1.7.5.jar to the same location.

10. Start JBoss by running the command JBOSS_HOME/bin/standalone.

11. Examine the log file JBOSS_HOME/standalone/log/server.log to be certain the Guidewire Solr Extension web application started successfully.

The application started successfully if you see a number of log entries related to the Guidewire Solr home directory.

12. In a browser, open the administrative user interface for the Guidewire Solr Extension web application by entering the following URL:

```
http://hostName:8983/pc-gwsolr
```

13. Verify all the following in the Guidewire Solr Extension administrative user interface.

a. You see links to administrative pages for each entity type that is searchable in PolicyCenter with free-text search. For example, the link to the administrative page for policies looks like:

Admin pc_policy_active

b. On the administrative page, you can enter a query and adjust settings.

14. Proceed to “Setting Up the Free-text Batch Load Command” on page 100.

See also

- “Free-text Search System Architecture” on page 358 in the *Configuration Guide*

Setting Up Free-text Search for Tomcat

PolicyCenter free-text search requires that you set up an instance of Tomcat separate from the instance that runs your PolicyCenter application.

To set up free-text search for Tomcat

1. On the host where the Tomcat instance for free-text search resides, create an installation directory to use as the Guidewire Solr home directory. For example:
 - **On Unix** – /opt/gwsolr
 - **On Windows** – C:\opt\gwsolr
 This guide uses /opt/gwsolr as the Guidewire Solr home directory name.
2. Create the environment variable GWSOLR_HOME for the directory that you created in step 1.
3. Create a new opt/gwsolr/pc directory and extract the file PolicyCenter/solr/pc-gwsolr.zip to it from the host where your PolicyCenter application resides.
4. Copy the file /opt/gwsolr/pc/pc-gwsolr.xml to the directory TOMCAT_HOME/conf/Catalina/localhost.
5. On Windows, edit the file TOMCAT_HOME\conf\Catalina\localhost\pc-gwsolr.xml, and add the drive specifier to the front of the path in the docBase attribute, as the following example shows.

```
<Context docBase="C:\opt\gwsolr\pc\pc-gwsolr.war" debug="0" crossContext="true">
```

6. Edit the file TOMCAT_HOME/conf/server.xml, and change the connector port for the HTTP/1.1 protocol from 8080 to 8983, as the following example shows.

```
Connector port="8983" protocol="HTTP/1.1"
      connectionTimeout="20000"
      redirectPort="8443" />
```

The standard Solr port is 8983.

7. Check the web application deployment descriptor in autodeploy/pc-gwsolr/WEB-INF/web.xml for the definition of the solr.home environment variable:

```
<env-entry>
  <env-entry-name>solr/home</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>/opt/gwsolr/pc/solr</env-entry-value>
</env-entry>
```

If you installed the ZIP file in a directory other than /opt/gwsolr, you must now change the path in the <env-entry-value> XML element to match the actual install directory path.

8. Create a lib folder in GWSOLR_HOME\pc and place the following logging JAR and properties files there:

- jcl-over-slf4j-1.7.5.jar – Jakarta commons logging over SLF4J
- jul-to-slf4j-1.7.5.jar – Java Logging to SLF4J
- log4j.properties – Log4j properties
- log4j-1.2.16.jar – Log4j API
- slf4j-api-1.7.5.jar – SLF4J API
- slf4j-ext-1.7.5.jar – SLF4J Extensions
- slf4j-log4j12-1.7.5.jar – SLF4J to Log4j

Logging JAR files are provided in the logging_jars folder at the root of the pc-gwsolr.zip file.

IMPORTANT Install only the logging JARs and property file specified above.

9. Start Tomcat by running the command `TOMCAT_HOME/bin/startup`.

If you want to access the Guidewire Solr web application with a remote debugger, start Tomcat by running the following command.

```
TOMCAT_HOME/bin/catalina.sh jpda start
```

Then, you can connect a remote debugging session through port 8000.

10. Examine the log file `TOMCAT_HOME/logs/catalina` to ensure the Guidewire Solr Extension web application started successfully.

The application started successfully if you see a number of log entries related to the Guidewire Solr home directory.

11. In a browser, open the administrative user interface for the Guidewire Solr Extension web application by entering the following URL:

```
http://hostName:8983/pc-gwsolr
```

12. Verify all the following in the Guidewire Solr Extension administrative user interface.

- a. You see links to administrative pages for each entity type that is searchable in PolicyCenter with free-text search. For example, the link to the administrative page for policies looks like:

Admin pc_policy_active

- b. On the administrative page, you can enter a query and adjust settings.

13. Proceed to “Setting Up the Free-text Batch Load Command” on page 100.

See also

- “Free-text Search System Architecture” on page 358 in the *Configuration Guide*

Setting Up Free-text Search for WebLogic

PolicyCenter free-text search requires that you install an instance of WebLogic separate from the instance that runs your PolicyCenter application.

To set up free-text search for WebLogic

1. Create a WebLogic domain/instance running under JDK 1.7.
2. On the host where the WebLogic instance for free-text search resides, create an installation directory to use as the Guidewire Solr home directory. For example:
 - **On Unix** – `/opt/gwsolr`
 - **On Windows** – `C:\opt\gwsolr`

This guide uses `/opt/gwsolr` as the directory name.

3. Create a new `opt/gwsolr/pc` directory and extract the file `PolicyCenter/solr/pc-gwsolr.zip` to it from the host where your PolicyCenter application resides.
4. Within the `autodeploy` directory in the WebLogic instance, create a subdirectory `pc-gwsolr`. In that subdirectory, extract the contents of the `pc-gwsolr.war`.
5. Check the web application deployment descriptor in `autodeploy/pc-gwsolr/WEB-INF/web.xml` for the definition of the `solr.home` environment variable:

```
<env-entry>
  <env-entry-name>solr/home</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>/opt/gwsolr/pc/solr</env-entry-value>
</env-entry>
```

If you installed the ZIP file in a directory other than `/opt/gwsolr`, you must now change the path in the `<env-entry-value>` XML element to match the actual install directory path.

6. Create a `lib` folder in `GWSOLR_HOME\pc` and place the following logging JAR file there:

- `jcl-over-slf4j-1.7.5.jar` – Jakarta commons logging over SLF4J

The logging JAR file is provided in the `logging_jars` folder at the root of the `pc-gwsolr.zip` file.

IMPORTANT Install only the logging JAR specified above.

7. Copy the logging JAR `jcl-over-slf4j-1.7.5.jar` to the `WEBLOGIC_DOMAIN_DIR/lib` folder.

8. Start the WebLogic instance by running the command `WEBLOGIC_DOMAIN_DIR/startWebLogic`.

9. Proceed to “Setting Up the Free-text Batch Load Command” on page 100.

See also

- “Free-text Search System Architecture” on page 358 in the *Configuration Guide*

Setting Up Free-text Search for WebSphere

PolicyCenter free-text search requires that you install an instance of WebSphere separate from the instance that runs your PolicyCenter application.

To set up free-text search for WebSphere

1. Install and configure WebSphere

2. On the host where the WebSphere instance for free-text search resides, create an installation directory to use as the Guidewire Solr home directory. For example:

- **On Unix** – `/opt/gwsolr`
- **On Windows** – `C:\opt\gwsolr`

This guide uses `/opt/gwsolr` as the directory name.

3. Create the environment variable `GWSOLR_HOME` for the directory that you created in step 1.

4. Create a new `opt/gwsolr/pc` directory and extract to it the file `PolicyCenter/solr/pc-gwsolr.zip` from the host where your PolicyCenter application resides.

5. On Windows, edit the file `GWSOLR_HOME\pc\pc-gwsolr.xml`, and add the drive specifier to the front of the path in the `docBase` attribute, as the following example shows.

```
<Context docBase="C:\opt\gwsolr\pc\pc-gwsolr.war" debug="0" crossContext="true">
```

6. Check the web application deployment descriptor in `autodeploy/pc-gwsolr/WEB-INF/web.xml` for the definition of the `solr.home` environment variable:

```
<env-entry>
  <env-entry-name>solr/home</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>/opt/gwsolr/pc/solr</env-entry-value>
</env-entry>
```

If you installed the ZIP file in a directory other than `/opt/gwsolr`, you must now change the path in the `<env-entry-value>` XML element to match the actual install directory path.

7. Create a `lib` folder in `GWSOLR_HOME\pc` and place the following logging JAR files there:

- `slf4j-api-1.7.5.jar` – SLF4J API
- `slf4j-ext-1.7.5.jar` – SLF4J Extensions
- `slf4j-jdk14-1.7.5.jar` – SLF4J Java logging binding
- `jcl-over-slf4j-1.7.5.jar` – Jakarta commons logging over SLF4J
- `log4j-over-slf4j-1.7.5.jar` – Log4j logging over SLF4J

Logging JAR files are provided in the `logging_jars` folder at the root of the `pc-gwsolr.zip` file.

IMPORTANT Install only the logging JARs specified above.

8. Start WebSphere:

- **On Unix** – `WEBSHPERE_HOME/bin startServer.sh server1`
- **On Windows** – **Start** → **All Programs** → **IBM WebSphere** → **Application Server** → **Profiles** → **AppSrv01** → **Start the server**

9. In a browser, open the WebSphere administrative console and log in:

`http://localhost:9060/ibm/console/login.do`

10. Change the port for the default host in your WebSphere application server and for its virtual host.

PolicyCenter assumes the Solr port is 8983. If you want to use a different port number, change the port number in the PolicyCenter file `solrserver-config.xml`.

- a. From the Administrative Console, navigate to **Servers** → **Server Types** → **WebSphere application servers** and click the name of your application server from the list. The default name of your application server is `server1`. The console displays the configuration page for your application server.
- b. On the right underneath **Communications**, click **Ports**.
- c. In the list of TCP/IP ports, click `WC_defaulthost`.
- d. In the Port field, change the value from 9080 to 8983.
- e. Click **Apply**.
- f. In the **Messages** box, Click **Save** to apply the changes to the master configuration.
- g. From the Administrative Console, navigate to **Environment** → **Virtual hosts** and click the name of your virtual host. The default name of your virtual host is `default_host`. The console displays the configuration page for your virtual host.
- h. On the right underneath **Additional Properties**, click **Host Aliases**.
- i. Click **New**. The console displays the configuration page for your new virtual host alias.
- j. Enter the following values.

Field name	Description
Host Name	Enter an asterisk (*).
Port	Enter 8983.

- k. Click **Apply**.
- l. In the **Messages** box, Click **Save** to apply the changes to the master configuration.

11. Install the Guidewire Solr Extension web application:

- a. Log in to the WebSphere administrative console.
- b. Navigate to **Applications** → **New application** and click **New Enterprise Application**.
- c. Under **Local file system**, click **Browse...**
- d. In the dialog, navigate to the file `/opt/gwsolr/pc/pc-gwsolr.war` and select it. Then, click **Open**.
- e. Click **Next**.

After a moment, a screen appears that asks how you want to install the application.

- f. Select the **Fast Path** radio button and then click **Next**.

After a moment, a screen appears with steps listed along the left.

- g. Click **Step 3**, to skip past **Step 1** and **Step 2** and click **Next**.

The console displays an application resource warning, ADMA8019E, which you can ignore safely.

- h. Click **Continue**.

The console displays the page for step 4, where you map web modules to virtual hosts.

- i. Click **Next** to accept the mapping of the `pc-gwsolr.war` web module to the `default_host` virtual host.

The console displays the page for step 5.

- j. In the **Context Root** field, enter `/pc-gwsolr`.

- k. Click **Next**, and on the page for step 6, click **Finish**.

WebSphere installs the Guidewire Solr Extension web application. The Administration Console displays messages while the installation process progresses. When the installation completes successfully, you see the following:

```
Application pc-solr_war installed successfully.
```

- l. Click **Save** to apply the changes to the master configuration.

12. From the Administrative Console, install the necessary shared libraries from the WAR file:

- a. In the left navigation, expand the **Environment** node and click **Shared Libraries**.

- b. Create a shared library, and then list all the logging JAR files preceded by `${USER_INSTALL_ROOT}`:

```
${USER_INSTALL_ROOT}\lib\s1f4j-api-1.7.5.jar
${USER_INSTALL_ROOT}\lib\s1f4j-ext-1.7.5.jar
${USER_INSTALL_ROOT}\lib\s1f4j-jdk14-1.7.5.jar
${USER_INSTALL_ROOT}\lib\jcl-over-s1f4j-1.7.5.jar
${USER_INSTALL_ROOT}\lib\log4j-over-s1f4j-1.7.5.jar
```

- c. Expand the **Applications** node. Click **Application Types** → **WebSphere Enterprise Applications**. In the right pane, click `pc-gwsolr` → **Shared Library References**.

- d. In the **Module** list, select `pc-gwsolr.war`. Click the **Reference Shared Libraries** button. Select the library you created in step b, and then click the right arrow button to move it from the **Available** list to the **Selected** list.

13. From the Administrative Console, change the class loading order

- a. Click **WebSphere Enterprise Applications** → `pc-gwsolr` → **Class loading and update detection** → **Detail Properties** tab.

- b. In the right pane, click **Class loading and update detection**.

- c. Modify class loading by selecting the **Classes loaded with local class loader first (parent last)** and **Single class loader for application** radio buttons.

- d. Click **OK**.

14. Log out of the administrative console and stop and start the application server.

15. Examine the log file `WEBSPPHERE_HOME/profiles/AppSrv01/logs/server1/SystemOut.log` to assure WebSphere initialized the Guidewire Solr web application successfully.

WebSphere initialized the application successfully if you see a log entries similar to the following:

```
[timeStamp] 00000009 SolrResourceL I org.apache.solr.core.SolrResourceLoader locateSolrHome
No /solr/home in JNDI
[timeStamp] 00000009 SolrResourceL I org.apache.solr.core.SolrResourceLoader locateSolrHome using
system property solr.solr.home: /opt/gwsolr/pc/solr
[timeStamp] 00000009 SolrServlet I org.apache.solr.servlet.SolrServlet init SolrServlet.init() done
[timeStamp] 00000009 servlet I com.ibm.ws.webcontainer.servlet.ServletWrapper init SRVE0242I:
[pc-gwsolr_war] [/pc-gwsolr] [SolrServer]: Initialization successful.
```

16. In a browser, open the administrative user interface for the Guidewire Solr Extension web application by entering the following URL:
`http://hostName:8983/pc-gwsolr`
17. Verify all the following in the Guidewire Solr Extension administrative user interface.
 - a. You see links to administrative pages for each entity type that is searchable in PolicyCenter with free-text search. For example, the link to the administrative page for policies looks like:
`Admin pc_policy_active`
 - b. On the administrative page, you can enter a query and adjust settings.
18. Proceed to “Setting Up the Free-text Batch Load Command” on page 100.

See also

- “Free-text Search System Architecture” on page 358 in the *Configuration Guide*

Setting Up the Free-text Batch Load Command

Follow these instructions only after you successfully set up free-text search for JBoss, Tomcat, WebLogic, or WebSphere. On Windows, ensure a Unix-compatible sort binary that performs character-value sorting is available. Guidewire supports cygwin on Windows.

Note: Do not set up and configure the free-text batch load command if you configure free-text search for embedded operation. Instead, use the Solr Data Import batch process.

The configuration files that you modify during set up, including the batch load command itself, are located in the following directory:

`/opt/gwsolr/pc/solr/policy_active/conf`

Most of the setup information for the free-text batch load command is located in a configuration file for your database brand. The configuration filename has the format:

`batchload-config-databaseBrand.xml`

The configuration parameters that you may need to modify include:

- `xsi:noNamespaceSchemaLocation` – Location of the XSD for batch load configuration files. You must modify this parameter if your Guidewire Solr home directory is other than `/opt/gwsolr`.
- `absolutePathToWorkDir` – Location of a working directory, possibly on a remote host, for collating and compiling large volumes of index documents for the full-text search engine. Generally in a production environment, you modify this parameter. Otherwise, the working directory is within the Guidewire Solr home directory. Assure the directory you specify has sufficient high-performance disk space.
- `absolutePathToSortTmpDir` – Location of a working directory into which the sort binary writes its intermediate files. This directory may be the same as or different from the directory specified by `absolutePathToWorkDir`.
- `absolutePathToSortExe` – Location of a sort binary. You must modify this parameter if you use a binary other than `/bin/sort` on Unix or `c:\cygwin\bin\sort.exe` on Windows.
- `dataSource` – Connection information for the PolicyCenter relational database. You must modify this parameter to specify the network location of the database and the username and password.
- `absolutePathToPostprocessorExe` – Locates the `postprocess` shell script of batch file that the batch load command calls after extracting data from the PolicyCenter relational database. You must modify this parameter if your Guidewire Solr home directory is other than `/opt/gwsolr`.

To set up the free-text batch load command

1. Navigate to the Guidewire Solr home directory.

2. Extract the files from the Guidewire Solr WAR file as you would any ZIP file, to the following location, using full directory paths:

```
opt/gwsolr/pc/exploded
```

The free-text batch load command depends on access to JAR files within the WAR file.

3. Switch to the configuration directory, `opt/gwsolr/pc/solr/policy_active/conf`.
4. Edit the `batchload` shell script or batch file, and modify the `CONFIGFILE` environment variable to locate the appropriate batch load configuration file for your database brand. For example, if your database brand is Oracle:

On Unix

```
CONFIGFILE=$GWSOLR_HOME/solr/policy_active/conf/batchload-config-oracle.xml
```

On Windows

```
set CONFIGFILE=%GWSOLR_HOME%\solr\policy_active\conf\batchload-config-oracle.xml
```

5. Open the batch load configuration file that you specified in step 4 and do all of the following:

- a. If you use a directory for the Guidewire Solr home directory other than `/opt/gwsolr`, modify the `xsi:noNamespaceSchemaLocation` attribute of the `<document>` element.
- b. Review the following elements for possible changes.

On Unix

```
<param name="absolutePathToWorkDir" value="/opt/gwsolr/pc/solr/policy_active/conf/loadDir"/>
<param name="absolutePathToSortExe" value="/bin/sort"/>
```

On Windows

```
<param name="absolutePathToWorkDir" value="c:\opt\gwsolr\pc\solr\policy_active\conf\loadDir"/>
<param name="absolutePathToSortExe" value="c:\cygwin\bin\sort.exe"/>
```

- c. Modify the attributes of the `<dataSource>` element to match the values for your database. For example, if your database brand is Oracle:


```
<dataSource name="ds_orcl" driver="oracle.jdbc.driver.OracleDriver"
url="jdbc:oracle:thin:@grinch:11201:gwDiaAsc" user="su" password="gw"/>
```
- d. If you use a directory for the Guidewire Solr home directory other than `/opt/gwsolr`, modify the `absolutePathToPostprocessorExe` attribute of the `<postprocessor>` element.
6. On Windows, edit `postprocess.bat` and modify the `CoreUtils` environment variable to locate the sort binary. For example:


```
set CoreUtils C:\opt\cygwin
```
7. If use a directory for the Guidewire Solr home directory other than `/opt/gwsolr`:
 - a. Open the file `data-config.xml`.
 - b. Change the value of the `url` attribute on the `<entity>` element to match the location of your Guidewire Solr home directory. For example:

```
<entity name="policy"
processor="XPathEntityProcessor"
orEach="/CONTAINER_ELEM/POLICY"
url="/opt/gwsolr/pc/solr/policy_active/conf/loadDir/policy"
stream="true">
```

8. Restart the application server to pick up the changes.
9. Run the `batchload` command to test the setup.
10. Examine the status response to verify your setup.

A problem-free load gives the same positive counts for Total Rows Fetched and Total Documents Processed.
11. Proceed to “Enabling Free-text Search in PolicyCenter” on page 362 in the *Configuration Guide*.

See also

- “Free-text Batch Load Command” on page 189 in the *System Administration Guide*

- “Configuring the Free-text Batch Load Command” on page 369 in the *Configuration Guide*
- “Free-text Search System Architecture” on page 358 in the *Configuration Guide*

Changing the Superuser Password

PolicyCenter 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 PolicyCenter for the first time, log in to PolicyCenter 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/pc/PolicyCenter.do`
For example, if connecting on your local computer, use:
`http://localhost:8080/pc/PolicyCenter.do`
3. Log into PolicyCenter 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 22 in the *System Administration Guide*.

At this point, you have a running PolicyCenter server. However, there is no data in the installation and you have none of the tools available to manage the PolicyCenter server process. Before you use PolicyCenter, follow the processes defined in “Generating Java and SOAP API Libraries” on page 102, and “Connecting to PolicyCenter with a Web Client” on page 117.

Generating Java and SOAP API Libraries

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

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

To generate the Java and SOAP APIs

1. From a command prompt, navigate to `PolicyCenter/bin`.
2. Execute the following command: `gwpc regen-java-api`.
This command generates the `java-api` directory within the top-level PolicyCenter directory.
3. Execute the following command: `gwpc regen-soap-api`.
This command generates the `soap-api` directory within the top-level PolicyCenter directory.

For more information, see “Integration Overview” on page 23 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 7.0.0 or newer. PolicyCenter publishes the following web services in the base configuration:

- `gw.webservice.pc.pc700.ccintegration.CCPolicySearchIntegration`

- `gw.webservice.pc.pc800.ccintegration.CCPolicySearchIntegration`

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



ClaimCenter publishes the following web services in the base configuration:

- `gw.webservice.cc.cc700.pcintegration.PCClaimSearchIntegrationAPI`
- `gw.webservice.cc.cc800.pcintegration.PCClaimSearchIntegrationAPI`

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

Note: These instructions describe how to set up ClaimCenter 8 to integrate with both PolicyCenter 8 and PolicyCenter 7.0.2 and later. They are followed by instructions describing how to integrate PolicyCenter 8 with ClaimCenter. If you are integrating ClaimCenter 8 with PolicyCenter 7.02 or later, use the instructions in the PolicyCenter 7 documentation for the PolicyCenter side of the integration.

1. Start the PolicyCenter server.
2. Launch ClaimCenter Studio. From a command prompt, navigate to the `bin` directory in your ClaimCenter installation, and type `gwcc studio`.
3. In the Studio Project window, navigate to **configuration** → **config** → **Plugins** → **registry**, and then open `IPolicySearchAdapter.gwp`.
4. Click **Remove**  to remove the demonstration implementation, `gw.plugin.policy.impl.PolicySearchPluginDemoImpl`.
5. Click **Add** .
6. Select **Add Gosu Plugin**.
7. Enter one of the following for the **Gosu Class** field, depending on the version of PolicyCenter that you are integrating with ClaimCenter:



PolicyCenter 7.0.2 or later: `gw.plugin.pcintegration.pc702.PolicySearchPCPlugin`

PolicyCenter 8.0.x: `gw.plugin.pcintegration.pc800.PolicySearchPCPlugin`

Note: The `pc702` plugin implementation class might indicate that it is deprecated. Ignore this indicator. This class is the correct one to use to integrate with PolicyCenter 7.0.2 and later.

8. If you are using the `pc702` plugin, add user name and password parameters.

Note: The `pc800` plugin uses a different technique for authentication. See step 16 for information on using the `PCConfigurationProvider` class.

- a. Next to **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. Next to **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`.
9. In the Studio Project window, navigate to **configuration** → **config** → **suite**, and then open `suite-config.xml`.
 10. Remove the comment markers `<!--` and `-->` from the line for the PolicyCenter URL:


```
<!--<product name="pc" url="http://localhost:8180/pc"/>-->
```


11. Update the PolicyCenter URL to match your server and port.
12. In the Studio **Project** window, navigate to **configuration** → **config**, and then open `config.xml`.
13. Set the PolicyCenter URL in the `PolicySystemURL` configuration parameter. For example, add the following line to this file:


```
<param name="PolicySystemURL" value="http://localhost:8180/pc"/>
```
14. In the Studio **Project** window, expand **configuration** → **gsrc** and then navigate to `wsi.remote.gw.webservice.pc`. Open one of the following web service files, depending on the version of PolicyCenter that you are integrating with ClaimCenter:

PolicyCenter 7.0.2 or later: `pc700.wsc`

PolicyCenter 8.0.x: `pc800.wsc`

Note: The `{pc}` variable for each of the defined web services is defined in `suite-config.xml`.
15. Ensure that the PolicyCenter server is running. Then select all the web services in **Resources** and click **Fetch Updates**. You must refresh the web services, even if you have made no changes to them.
16. In the same editor on the **Settings** tab is the configuration provider class. This class, `wsi.remote.gw.webservice.pc.PCConfigurationProvider`, is used by the `pc800` plugin to define the user name and password that ClaimCenter uses to connect with PolicyCenter.

In the base configuration, ClaimCenter defines the user name `su` and password `gw` in this class. Guidewire recommends that you change this user name in this class. Then add a new user to PolicyCenter with the same user name and password that you specify in `PCConfigurationProvider`. That new user must have the `soapadmin` permission, plus at least the permissions needed to view and edit policies.

There is an example of how to configure a user name and password with `ContactManager` that is similar to the ClaimCenter configuration for PolicyCenter. See “Configuring ClaimCenter-to-ContactManager Authentication” on page 74 in the *Contact Management Guide*.
17. To test the integration, restart the ClaimCenter server. Then start the **New Claim** wizard and search for a **Personal Auto** or **Workers' Compensation** policy. The integration with PolicyCenter 8.0 supports all base integration policy types except **Farmowner's**, **Professional Liability**, and **Personal Travel**.

The ClaimCenter plugin implements 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 provides different versions of this file corresponding to your version of PolicyCenter. In the **Project** window, navigate to **configuration** → **config** → **datamapping** → **pc**:

- **PolicyCenter 7.0.2 or later:** `702` → `pc-to-cc-data-mapping.xml`
- **PolicyCenter 8.0.x:** `800` → `pc-to-cc-data-mapping.xml`



The only object that ClaimCenter sends to PolicyCenter is a `PCSearchCriteria` object. This object is created and populated from the ClaimCenter search criteria by using Gosu. Configuration of this object 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, restart your ClaimCenter server.

To configure PolicyCenter to retrieve claim information from ClaimCenter

Note: These instructions describe how to set up PolicyCenter 8 to integrate with both ClaimCenter 8 and ClaimCenter 7. They are preceded by instructions describing how to integrate ClaimCenter 8 with PolicyCenter. If you are integrating PolicyCenter 8 with ClaimCenter 7, use the instructions in the ClaimCenter 7 documentation for the ClaimCenter side of the integration.

1. Start the ClaimCenter server.
2. Launch PolicyCenter Studio. From a command prompt, navigate to the bin directory in your PolicyCenter installation, and type `gwpc studio`.
3. In the Studio **Project** window, navigate to **configuration** → **config** → **suite**, and then open `suite-config.xml`.
4. Remove the comment markers `<!--` and `-->` from the line for the ClaimCenter URL:

```
<!--<product name="cc" url="http://localhost:8080/cc"/>-->
```
5. Update the ClaimCenter URL to match your server and port.
6. In the Studio **Project** window, expand **configuration** → **gsrc** and then navigate to `wsi.remote.gw.webservice.cc`. Open one of the following web service files, depending on the version of ClaimCenter that you are integrating with PolicyCenter:
ClaimCenter 7.0.x: `cc700.wsc`
ClaimCenter 8.0.x: `cc800.wsc`
 Note the `#{cc}` variable for the defined web service. This variable is defined in `suite-config.xml`.
7. Ensure that the ClaimCenter server is running. Then select the web service in **Resources** and click **Fetch Updates**. You must refresh the web service, even if you have made no changes to it.
8. In the same editor on the **Settings** tab is the configuration provider class. This class, `wsi.remote.gw.webservice.cc.CCConfigurationProvider`, defines the user name and password that PolicyCenter uses to connect with ClaimCenter.
 In the base configuration, PolicyCenter uses the user name `su` and password `gw`. Guidewire recommends that you change this user name and password in `CCConfigurationProvider` and then create a corresponding user with the same user name and password in ClaimCenter. That new user must have the `soapadmin` permission, plus at least the permissions needed to view and edit claims and policies.
 There is an example of how to configure a user name and password with ContactManager that is similar to the PolicyCenter configuration for ClaimCenter. See “Configuring PolicyCenter-to-ContactManager Authentication” on page 75 in the *Contact Management Guide*.
9. In the Studio **Project** window, navigate to **configuration** → **config** → **Plugins** → **registry**, and then open `IClaimSearchPlugin.gwp`.
10. Click **Remove**  to remove the demonstration implementation, `gw.plugin.claimsearch.impl.GWDemoClaimSearchPlugin`.
11. Click **Add** .
12. Select **Add Gosu Plugin**.
13. Enter one of the following plugin implementation classes in the **Gosu Class** field, depending on the version of ClaimCenter that you are integrating with PolicyCenter:
ClaimCenter 700: `gw.plugin.claimsearch.cc700.GWClaimSearchPlugin`
ClaimCenter 800: `gw.plugin.claimsearch.cc800.GWClaimSearchPlugin`
Note: The `cc700` plugin implementation class might indicate that it is deprecated. Ignore this indicator. This class is the correct one to use to integrate with ClaimCenter 7.0.0 and later.
14. In the **Project** window, navigate to **configuration** → **config**, and then open `config.xml`.

15. Set the `ClaimSystemURL` parameter to the ClaimCenter URL. Before configuration, the `ClaimSystemURL` is set to a null value. In the installation of the base configuration, the ClaimCenter URL is:
`http://localhost:8080/cc/ClaimCenter.do`
16. Restart PolicyCenter to pick up these changes.

See also

- “PolicyCenter Exit Points to ClaimCenter and BillingCenter” on page 512 in the *Integration Guide*

Enabling Large Loss Notification Integration

In the base configuration, ClaimCenter does not enable integration with a policy system for large loss notification.

Perform the following instructions in ClaimCenter Studio to enable this behavior with PolicyCenter. To successfully complete all the steps, the PolicyCenter server must be running. The ClaimCenter server does not need to be running. After performing these instructions, you must restart the ClaimCenter server for the changes to take effect.

To enable large loss notification integration between ClaimCenter 8.0.x and PolicyCenter 8.0.x

1. Launch ClaimCenter Studio.
 From a command prompt, navigate to the `bin` directory in your ClaimCenter installation, and type `gwcc studio`.
2. In the Studio **Project** window, navigate to `configuration` → `config` → `Rule Sets` → `Preupdate` → `TransactionSetPreupdate`.
3. Select the check box for the **Large Loss Notification** rule to enable that rule.
4. Navigate to `configuration` → `config` → `Rule Sets` → `EventMessage` → `EventFired` and select the check box for the **Policy System Notification** rule to enable that rule.
5. Navigate to `configuration` → `config` → `Messaging`, and then open `messaging-config.xml`.
6. In the editor, click the row for the messaging destination
`Java.MessageDestination.PolicySystemNotification.Name`, and then select the **Enabled** check box.
7. In the Studio **Project** window, navigate to `configuration` → `config` → `Plugins` → `registry`, and then open `policySystemNotificationTransport.gwp`.
8. In the editor, select the **Enabled** check box.
9. In the Studio **Project** window, navigate to `configuration` → `config` → `Plugins` → `registry`, and then open `IPolicySystemNotificationPlugin.gwp`. In the base implementation, ClaimCenter uses the plugin implementation `gw.plugin.policy.notification.pc800.PCPolicySystemNotificationPlugin`.
10. In the editor, select the **Enabled** check box.
11. If you have not already done so, update `suite-config.xml` to enable ClaimCenter to work with PolicyCenter. Also, if necessary, change the URL for the PolicyCenter server to the URL used in your configuration.
 - a. In the Studio **Project** window, navigate to `configuration` → `config` → `suite`, and then open `suite-config.xml`.
 - b. If present, remove the comment markers `<!--` and `-->` from the line defining the PolicyCenter URL:
`<!--<product name="pc" url="http://localhost:8180/pc"/>-->`
 - c. If necessary, update the PolicyCenter URL to match your server and port.
12. Verify that the PolicyCenter server is running.
13. In the ClaimCenter Studio **Project** window, navigate to `configuration` → `gsrsrc` and then to `wsi.remote.gw.webservice.pc`.

14. Double-click `pc800.wsc` to open this web service collection in the editor.
15. Select the web service `${pc}/ws/gw/webService/pc/pc800/ccintegration/ClaimToPolicySystemNotificationAPI?wsdl`.

Note: The `${pc}` part of this path is the PolicyCenter URL defined in `suite-config.xml`.
16. Click **Fetch Updates** to refresh the web service.
17. In the same editor on the **Settings** tab is the configuration provider class. This class, `wsi.remote.gw.webService.pc.PCConfigurationProvider`, defines the user name and password that ClaimCenter uses to connect with PolicyCenter.

For information on setting the user name and password in this class, see step 16 in the previous instructions describing how to configure ClaimCenter to retrieve policy information from PolicyCenter.
18. Restart the ClaimCenter server for these changes to take effect.

Test your integration

In a development system, test the integration by adding a claim to ClaimCenter with a loss that exceeds the threshold for that type of claim. If the integration works correctly, PolicyCenter adds a referral reason and an activity for that policy.

Enabling Integration between BillingCenter and PolicyCenter

This topic applies if you have both PolicyCenter and BillingCenter installed and want to integrate them to share information. If you are not integrating PolicyCenter with BillingCenter, skip this topic.

The default installations of PolicyCenter and BillingCenter support integration between the two applications. The integration enables PolicyCenter and BillingCenter to exchange information about accounts, policies, producers, producer codes, and billing. When the integration is enabled, the payment screen in PolicyCenter displays payment plans retrieved from BillingCenter. Your payment plan selection is transmitted to BillingCenter and saved with the policy period. Accounts and policy periods are shared between BillingCenter and PolicyCenter. BillingCenter sends delinquency notices to PolicyCenter. The PolicyCenter user interface displays links that enable you to view data in BillingCenter.

BillingCenter includes a Gosu plugin that interfaces with a specialized PolicyCenter web service provided with PolicyCenter 7.0.0 or newer.

PolicyCenter publishes the following web services in the base configuration for use by BillingCenter:

- `gw.webService.pc.pc800.job.PolicyRenewalAPI`
- `gw.webService.pc.pc800.job.CancellationAPI`
- `gw.webService.pc.pc700.job.PolicyRenewalAPI`
- `gw.webService.pc.pc700.job.CancellationAPI`

PolicyCenter includes Gosu plugins that interface with BillingCenter web services provided with BillingCenter. These plugins enable PolicyCenter to send information to and retrieve information from BillingCenter.

BillingCenter publishes the following web services in the base configuration for use by PolicyCenter:



- `gw.webService.policycenter.bc801.BillingAPI`
- `gw.webService.policycenter.bc801.BillingSummaryAPI`
- `gw.webService.bc.bc801.PaymentInstrumentAPI`
- `gw.webService.bc.bc801.BCAPI`
- `gw.webService.policycenter.bc700.BillingAPI`
- `gw.webService.policycenter.bc700.BillingSummaryAPI`
- `gw.webService.bc.bc700.PaymentInstrumentAPI`

When PolicyCenter starts, it sends `Producer`, `ProducerCode`, `Account`, and `Policy` entity instances to BillingCenter. Therefore, you must start BillingCenter before PolicyCenter.

For detailed information, see “Billing Integration” on page 455 in the *Integration Guide* and “Billing System Integration” on page 733 in the *Application Guide*.

To enable the PolicyCenter plugin in BillingCenter

1. In a command prompt, navigate to the *BillingCenter/bin* directory in the BillingCenter installation.
2. Type the following command:

```
gwbc studio
```
3. In Studio, navigate in the Project window to `configuration` → `config` → `Plugins` → `registry`, and then open `IPolicySystemPlugin.gwp`.
4. Change `Gosu Class` to `gw.plugin.pas.pc800.PCPolicySystemPlugin`.
5. Add user name and password parameters.
 - a. Next to **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. Next to **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`.
6. In the Project window, navigate to `configuration` → `config` → `suite` and double-click `suite-config.xml`.
7. Remove the comment markers `<!--` and `-->` from the line for the PolicyCenter URL:


```
<!--<product name="pc" url="http://localhost:8180/pc"/>-->
```
8. If necessary, update the PolicyCenter URL to match your server and port.
9. In the Studio Project window, expand `configuration` → `gsrc` and then navigate to `wsi.remote.gw.webservice.pc`. Open one of the following web service collection files, depending on the version of PolicyCenter that you are integrating with BillingCenter:
 - `pc700.wsc`
 - `pc800.wsc`

Note: The `${pc}` variable for each of the defined web services is defined in `suite-config.xml`.
10. Ensure that the PolicyCenter server is running.
11. Select all the web services in **Resources** and click **Fetch Updates**. You must refresh the web services, even if you have made no changes to them.
12. Shut down the PolicyCenter server.
13. Click **File** → **Save All**.

To start BillingCenter

1. In a command prompt, navigate to the *BillingCenter/bin* directory in the BillingCenter installation.
2. Type the following command:

```
gwbc dev-start
```

3. After BillingCenter is ready, open a supported web browser and enter the URL to BillingCenter. In the default configuration, the web address is:
`http://localhost:8580/bc/BillingCenter.do`
4. Log in as user `su` with password `gw`.
5. For testing, development, or demonstration purposes, load sample data.
 - a. Press ALT+SHIFT+T to open the **Server Tools** screen.
 - b. Click **Sample Data** in the sidebar menu.
 - c. On the **Sample Data** screen, click **Import**.
 - d. After the data set loads, in the **Options** menu , click **Return to BillingCenter**.

To enable BillingCenter plugins in PolicyCenter

1. At a command prompt, navigate to the `PolicyCenter/bin` directory in the PolicyCenter installation.
2. Start Studio for PolicyCenter by using the following command:
`gwpc studio`
3. In the **Project** window in Studio, navigate to **configuration** → **config** → **Plugins** → **registry**, and then open `IBillingSummaryPlugin.gwp`.
4. Change **Gosu Class** to one of the following, depending on the version of BillingCenter that you are integrating:
`gw.plugin.billing.bc800.BCBillingSummaryPlugin`
`gw.plugin.billing.bc700.BCBillingSummaryPlugin`
5. In the **Project** window in Studio, navigate to **configuration** → **config** → **Plugins** → **registry**, and then open `IBillingSystemPlugin.gwp`.
6. Change **Gosu Class** to one of the following, depending on the version of BillingCenter that you are integrating:
`gw.plugin.billing.bc800.BCBillingSystemPlugin`
`gw.plugin.billing.bc700.BCBillingSystemPlugin`
7. In the **Project** window, navigate to **configuration** → **config**, and then open `config.xml`.
8. Set the `BillingSystemURL` parameter to the BillingCenter URL. Before configuration, the `BillingSystemURL` is set to a null value. In a default installation, the BillingCenter URL is:
`http://localhost:8580/bc/BillingCenter.do`

Note: This configuration parameter defines the URL for an exit point. Exit points are configured in `config.xml` and not in `suite-config.xml`. The configuration parameters in `suite-config.xml` support integration between Guidewire applications through web services. The exit point configuration parameters in `config.xml` support integration between web browsers.

See “PolicyCenter Exit Points to ClaimCenter and BillingCenter” on page 512 in the *Integration Guide*.
9. In the Studio **Project** window, navigate to **configuration** → **config** → **suite**, and then open `suite-config.xml`.
10. Remove the comment markers `<!--` and `-->` from the line for the BillingCenter URL:
`<!--<product name="bc" url="http://localhost:8580/bc"/>-->`
11. If necessary, update the BillingCenter URL to match your server and port.
12. In the Studio **Project** window, expand **configuration** → **gsrsc** and then navigate to `wsi.remote.gw.webservice.bc`. Open one of the following web service collection files, depending on the version of BillingCenter that you are integrating with PolicyCenter:
 - `bc700.wsc`
 - `bc800.wsc`


Note: Note: The `${bc}` variable for each of the defined web services is defined in `suite-config.xml`.

13. Ensure that the BillingCenter server is running. Then, in PolicyCenter Studio, select all the web services in **Resources** and click **Fetch Updates**. You must refresh the web services, even if you have made no changes to them.
14. In the same editor on the **Settings** tab is the configuration provider class. This class, `wsi.remote.gw.webservice.bc.BCConfigurationProvider`, is used by the bc700 and bc800 plugins to define the user name and password that PolicyCenter uses to connect with BillingCenter.
 In the base configuration, PolicyCenter defines the user name `pu` and password `gw` in this class. Guidewire recommends that you change the user name and password defined in this class. Then add a new user to BillingCenter with the same user name and password that you specify in `BCConfigurationProvider`. That new user must have the `soapadmin` permission, plus at least the permissions needed to view policies and accounts.
 There is an example of how to configure a user name and password with ContactManager that is similar to the PolicyCenter configuration for BillingCenter. See “Configuring PolicyCenter-to-ContactManager Authentication” on page 75 in the *Contact Management Guide*.
15. Click **File** → **Save All**.

To start PolicyCenter

1. Before starting PolicyCenter, first start BillingCenter if it is not already running. See “To start BillingCenter” on page 108.
 If PolicyCenter is integrated with BillingCenter, on startup, PolicyCenter sends `Producer`, `ProducerCode`, `Account`, and `Policy` entity instances to BillingCenter. Therefore, you must start BillingCenter before PolicyCenter, so BillingCenter is available to receive this data.
2. In a command prompt, navigate to the `PolicyCenter/bin` directory in the PolicyCenter installation.
3. Start the PolicyCenter server by entering the following command:

```
gwpc dev-start
```
4. After PolicyCenter is ready, open a supported web browser and navigate to PolicyCenter. In the default installation, the web address is:

```
http://localhost:8180/pc/PolicyCenter.do
```
5. For testing, development or demonstration purposes, load sample data.
 - a. Log in as user `su` with password `gw`.
 - b. Type `ALT+SHIFT+T`.
 - c. Click the **Internal Tools** tab.
 - d. Click **PC Sample Data** in the sidebar.
 - e. Click **Load** to load one of the sample data sets.
 - f. After the data set loads, in the **Options** menu , click **Return to PolicyCenter**.

Verifying the BillingCenter Integration

When you enabled the integration, PolicyCenter sent `Producer`, `ProducerCode`, `Account`, and `Policy` entity instances to BillingCenter. These instructions show you how to verify that BillingCenter has these entity instances.

To verify the integration

1. Verify that policies were transferred from PolicyCenter to BillingCenter.
 - a. In PolicyCenter, navigate to **Search** tab → **Policies** and click **Policies**.

- b. In the **Search Policies** screen, enter enough data to search for a policy.
For example, if the small sample data set is loaded, enter Ray in the **First Name** field, Newton in the **Last Name** field, and 100-002541 in the **Producer Code** field.
 - c. Select a policy number and copy it.
 - d. In BillingCenter, click the drop-down button on the **Policy** tab, and then paste the policy number in the **Policy #** box.
 - e. Click **Search**. BillingCenter displays a list of matching policies.
 - f. In the **Policy** column, click the link to the policy. BillingCenter displays the **Summary** page for that policy.
2. Verify that accounts were transferred from PolicyCenter to BillingCenter.
 - a. In PolicyCenter, select an account number and copy it.
 - b. In BillingCenter, select the drop-down button on the **Account** tab, and paste the account number in the **Account #** box.
 - c. Click **Search**. BillingCenter displays a list of matching accounts.
 - d. In the **Account** column, click the link to the account. BillingCenter displays the **Summary** page for that account.
 3. Verify that producers and producer codes were transferred from PolicyCenter to BillingCenter.
 - a. In PolicyCenter, click an account number to jump to the **Account File Summary** page for the account.
 - b. Copy the **Producer Code** on that account.
 - c. In BillingCenter, select **Search → Producers**.
 - d. Paste the producer code in the **Producer Code** field.
 - e. Click **Search**.
BillingCenter displays a list of matching producers. In the **Name** column, click the link to the producer to see the producer **Summary** page.

Enabling Archiving or Disabling Archiving Work Queues

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 and restore work queues.

To disable the archive and restore work queues

1. In a command window, navigate to the `PolicyCenter\bin` directory in the PolicyCenter installation.
2. Launch Guidewire Studio using the following command:
`gwpc studio`
3. In the **Project** window, navigate to **configuration → config → workqueue**, and then open `work-queue.xml`.
4. Comment out the following block by adding `<!--` before the block and `-->` after it:

```
<work-queue workQueueClass="com.guidewire.pc.domain.archive.ArchivePolicyTermWorkQueue"
progressInterval="600000">
```

```

        <worker instances="10"/>
    </work-queue>
    <work-queue workQueueClass="com.guidewire.pc.domain.archive.RestorePolicyTermWorkQueue"
        progressInterval="600000">
        <worker instances="10"/>
    </work-queue>

```

5. Save your changes.

See also

- “More Information on Archiving” on page 329 in the *Application Guide* for a list of topics related to archiving.

IMPORTANT Guidewire strongly recommends that you contact Customer Support before implementing archiving.

Installing Rating Management

Guidewire Rating Management is available within PolicyCenter. However, Rating Management is licensed separately from PolicyCenter. Contact your Guidewire sales representative for information on how to obtain Rating Management. Contact your Guidewire support representative for instructions on how to enable Rating Management.

After you have enabled Rating Management, you must enable the rating plugin that works with Rating Management.

See also

- “Guidewire Rating Management and PCRatingPlugin” on page 357 in the *Integration Guide*
- “Rating Management Concepts” on page 543 in the *Application Guide*

Installing Reinsurance Management or Disabling Work Queue

Guidewire Reinsurance Management is available within PolicyCenter. However, Reinsurance Management is licensed separately from PolicyCenter. Contact your Guidewire sales representative for information on how to obtain Reinsurance Management. Contact your Guidewire support representative for instructions on how to enable Reinsurance Management.

See “Reinsurance Management Concepts” on page 617 in the *Application Guide*.

If you do not enable Reinsurance Management, Guidewire recommends that you disable `RICedingWorkQueue`.

To disable `RICedingWorkQueue`

1. In a command window, navigate to the `PolicyCenter/bin` directory in the PolicyCenter installation.
2. Launch Guidewire Studio using the following command:

```
gwpc studio
```

3. In Studio, open `workqueue` → `work-queue.xml`.
4. Comment out the following block by adding `<!--` before the block and `-->` after it.

```

    <work-queue workQueueClass="com.guidewire.pc.domain.reinsurance.RICedingWorkQueue"
        progressInterval="30000">
        <worker instances="1"/>
    </work-queue>

```


5. Save your changes.

Upgrading Product Model of Old LOB Extension Packs

If you need to add a pre-8.0.4 Line of Business extension pack to your PolicyCenter 8.0.4 installation, you can upgrade the extension pack. The `gwpc upgrade-productmodel-to-structure` command upgrades product model files to the 8.0.4 structure without requiring a full configuration upgrade of PolicyCenter. The command also upgrades audit schedule pattern classes and code identifiers.

To upgrade the extension pack

1. Make a backup copy of the product model directory. The command destructively modifies the target directory.
2. Open a command line window to `PolicyCenter\bin`.
3. Run the `gwpc upgrade-productmodel-to-structure -DtargetPath=target_path` command. The `target_path` is the path to the configuration directory or other directory containing the config folder.

Configuring Single Sign-on Authentication

You can configure PolicyCenter to use single sign-on (SSO) authentication. PolicyCenter then forwards users from the application URL to an authentication provider which checks the users credentials and then forwards back to PolicyCenter. PolicyCenter generates a unique Cross-Site Request Forgery (CSRF) token for each user session. The CSRF token is included in each request and used by the server to verify the legitimacy of the user request.

The following configuration is a basic example. You can use this example to develop more complicated authentication features, such as redirecting users to different failure pages depending on the failure reason and so forth.

To configure SSO authentication

1. Create a Gosu class `CustomAuthServlet.gs`.
 - a. Open Studio.
 - b. In the Studio Project window, expand **configuration**.
 - c. Right-click `gsrc` and click **New** → **Package**.
 - d. Enter a package name for upgrade purposes, such as `companyName.auth`.
 - e. Right-click the package and click **New** → **Gosu Class**.
 - f. Enter `CustomAuthServlet` as the name for the class and click **OK**.
 - g. Enter the following class definition:

```
package companyName.auth

uses com.guidewire.pl.system.dependency.PLDependencies
uses com.guidewire.pl.system.service.context.ServiceToken
uses com.guidewire.pl.system.server.Version
uses com.guidewire.pl.web.controller.WebServlet
uses javax.servlet.http.HttpServletResponse
uses javax.servlet.http.HttpServletRequest
uses javax.servlet.http.HttpServlet
uses gw.servlet.ServletUtils
uses javax.security.auth.login.LoginException
uses gw.servlet.Servlet
uses gw.plugin.Plugins
uses gw.plugin.baseurlbuilder.IBaseURLBuilder
```

```

@Servlet( \ path : String ->path.matches( "/ssosaml" ) )
class CustomAuthServlet extends HttpServlet {
    override function doPost(req: HttpServletRequest, resp: HttpServletResponse) {
        var user:User = ServletUtils.getAuthenticatedUser(req, true);
        if (user != null) {
            redirectToIndex(req, resp);
            return;
        }

        // try to login
        try {
            PLDependencies.LoginManager.login(req);
        } catch (e : LoginException) {
            respondUnauthorized(req, resp);
            return;
        }

        var serviceToken:ServiceToken = PLDependencies.CommonDependencies.ServiceToken;
        if (serviceToken == null || !serviceToken.AuthenticatedUser) {
            respondUnauthorized(req, resp);
        } else {
            // store token
            req.getSession(false).setAttribute(WebServlet.SERVICE_TOKEN_SESSION_ATTR, serviceToken);
            redirectToIndex(req, resp);
        }

        return;
    }

    private function respondUnauthorized(req:HttpServletRequest, resp:HttpServletResponse) {
        print("User is unauthorized")
        redirectToError(req, resp);
    }

    private function redirectToIndex(req:HttpServletRequest, resp:HttpServletResponse) {
        print("User is authorized. Send to index page.")
        var plugin:IBaseURLBuilder = (IBaseURLBuilder) Plugins.get("BaseURLBuilderPlugin");
        var pcStartupPageEP = "PolicyCenterStartupPageEP"
        resp.sendRedirect(plugin.getApplicationBaseURL(req) + "/" + pcStartupPageEP + ".do");
    }

    private function redirectToError(req:HttpServletRequest, resp:HttpServletResponse) {
        print("User is unauthorized. Send to Default Failure page.")
        var plugin:IBaseURLBuilder = (IBaseURLBuilder) Plugins.get("BaseURLBuilderPlugin");
        var defaultFailureEP = "DefaultFailureEP"
        resp.sendRedirect(plugin.getApplicationBaseURL(req) + "/" + defaultFailureEP + ".do");
    }
}

```

2. Expand configuration → config → servlets and open servlets.xml.

3. Add your custom servlet to the list. For example:

```
<servlet class="companyName.auth.CustomAuthServlet"/>
```

4. Add an AuthServicePlugin.gosu class to your custom authentication package.

```

package companyName.auth

uses gw.plugin.security.AuthenticationServicePlugin
uses gw.plugin.security.AuthenticationServicePluginCallbackHandler
uses gw.plugin.security.AuthenticationSource
uses gw.plugin.security.UserNamePasswordAuthenticationSource
uses java.lang.IllegalArgumentException
uses javax.security.auth.login.FailedLoginException

class AuthServicePlugin implements AuthenticationServicePlugin {
    var _handler: AuthenticationServicePluginCallbackHandler;
    override function authenticate(p0: AuthenticationSource): String {
        if (p0.typeis UserNamePasswordAuthenticationSource == false) {
            throw new IllegalArgumentException("Authentication source type " + p0.getClass().getName() +
                "is not known to this plugin");
        }
        var uNameSource:UserNamePasswordAuthenticationSource = (UserNamePasswordAuthenticationSource) p0 ;
        var username = uNameSource.Username;
        var userPublicId = _handler.findUser(username);
        if (userPublicId == null) { throw new FailedLoginException("Bad user name " + username);}
        return userPublicId;
    }

    override function setCallback(p0: AuthenticationServicePluginCallbackHandler) {

```

```

        _handler = p0;
    }
}

```

5. Add an `AuthSourceCreator.gs` Gosu class to your custom authentication package.

```

package companyName.auth

uses gw.plugin.security.AuthenticationSourceCreatorPlugin
uses gw.plugin.security.AuthenticationSource
uses javax.servlet.http.HttpServletRequest
uses gw.plugin.security.UserNamePasswordAuthenticationSource

class AuthSourceCreator implements AuthenticationSourceCreatorPlugin {
    override function createSourceFromHttpRequest(p0: HttpServletRequest): AuthenticationSource {

        var source:AuthenticationSource;
        var userName:String = p0.getParameter("username");
        var password:String = p0.getParameter("password");





        print("userName\t" + userName)
        print("password\t" + password)

        source = new UserNamePasswordAuthenticationSource(userName, password);

        return source;
    }
}

```

In your code, check for errors and throw `InvalidAuthenticationSourceData` if there are errors.

6. Associate `AuthServicePlugin.gs` to the `AuthenticationServicePlugin` plugin.
 - a. Expand **configuration** → **config** → **Plugins** → **registry** and open `AuthenticationServicePlugin.gwp`.
 - b. Click  to remove the default plugin.
 - c. Click  and select **Add Gosu Plugin**.
 - d. For **Gosu Class** enter the `AuthServicePlugin.gs` class, including the fully qualified package.
7. Associate `AuthSourceCreator.gs` to the `AuthenticationSourceCreatorPlugin` plugin.
 - a. Open `AuthenticationSourceCreatorPlugin.gwp`.
 - b. Click  to remove the default plugin.
 - c. Click  and select **Add Gosu Plugin**.
 - d. For **Gosu Class** enter the `AuthSourceCreator.gs` class, including the fully qualified package.
8. Create an entry point for the entry page.
 - a. Expand **configuration** → **config** → **Page Configuration** → **pcf**, right-click **entrypoints** and click **New** → **PCF** file.
 - b. Enter `PolicyCenterStartupPageEP` for the file name.
 - c. Select **Entry Point** for the file type and click **OK**.
 - d. Select the entry point.
 - e. Set **location** to `PolicyCenterStartupPage()`.
 - f. Set **authenticationRequired** to `false`.
9. Create an entry point for the default failure page.
 - a. Right-click **entrypoints** and click **New** → **PCF** file.
 - b. Enter `DefaultFailureEP` for the file name.
 - c. Select **Entry Point** for the file type and click **OK**.

- d. Select the entry point.
 - e. Set `location` to `DefaultFailurePage()`.
 - f. Set `authenticationRequired` to `false`.
10. Create a `BaseURLBuilderPlugin`.
- a. Right-click `configuration` → `config` → `Plugins` → `registry` and click `New` → `Plugin`.
 - b. Enter `BaseURLBuilderPlugin` for the name.
 - c. Enter `IBaseURLBuilder` for the interface and click `OK`.
 - d. Click  and select `Add Java Plugin`.
 - e. Enter `com.guidewire.pl.web.render.html.BaseURLBuilderImpl` for the `Java Class`.
11. Save all changes.
12. Include the following form on an HTML page for testing.
- ```
<form name="input" action="http://localhost:8080/pc/service/ssosaml" method="post">
 Username: <input type="text" name="username">
 Password: <input type="text" name="password">

 <input type="submit" value="Submit">
</form>
```

## Starting PolicyCenter on the Application Server

Select the topic below for your application server type to learn how to start PolicyCenter:

- “Starting PolicyCenter on JBoss” on page 116
- “Starting PolicyCenter on Tomcat on Windows” on page 116
- “Starting PolicyCenter on WebLogic” on page 117
- “Starting PolicyCenter on WebSphere” on page 117

PolicyCenter can run in development, test, or production mode and at different run levels. For more information, see “Server Modes and Run Levels” on page 58 in the *System Administration Guide*.

### Starting PolicyCenter on JBoss

PolicyCenter 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 PolicyCenter on Tomcat on Windows

PolicyCenter 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 PolicyCenter on WebLogic

### To start PolicyCenter 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 PolicyCenter.
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 PolicyCenter application. WebLogic identifies PolicyCenter by the name that you gave to the deployed EAR file.
8. Click **Deploy Application**.

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

## Starting PolicyCenter on WebSphere

### To start PolicyCenter on WebSphere

1. Open the WebSphere Administrative Console.
2. Click **Applications** → **Application Types** → **WebSphere enterprise applications**.
3. Select the checkbox next to the PolicyCenter application, abbreviated by default as **pc**.  
If PolicyCenter is already running, and you want to restart PolicyCenter, restart WebSphere.  
If PolicyCenter is not running, click **Start**. PolicyCenter might take a few minutes to start.

---

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

---

## Connecting to PolicyCenter with a Web Client

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

`http://appserver1:8080/pc/PolicyCenter.do`

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

QuickStart	8180
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 43 for a list of required software and hardware for client computers accessing PolicyCenter.

## Configuring Windows Accessibility for Firefox

When a user clicks in a cell in an editable list view, the dotted border around actionable elements, such as links, cells and radio buttons, is not visible. However, if the user tabs into the same list view, the dotted border is visible. Firefox checks a Windows accessibility setting to determine this behavior.

### To always enable the dotted border around actionable elements

1. Click **Start** → **Control Panel** → **Ease of Access Center** → **Make the keyboard easier to use**.
2. Click **Underline keyboard shortcuts and access keys**.
3. Click **OK**.
4. Restart Firefox.

## Additional Installation Information

The following topics link to sources for more information.

### Integrating PolicyCenter with ContactManager

To integrate PolicyCenter with ContactManager, see “Integrating ContactManager with Guidewire Core Applications” on page 49 in the *Contact Management Guide*.

### Running PolicyCenter in a Clustered Environment

Running PolicyCenter in a clustered environment requires an in depth understanding of PolicyCenter configuration files. See “Clustering Application Servers” on page 77 in the *System Administration Guide*.

# Commands Reference

This topic lists the commands that are used to operate PolicyCenter 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.

This topic includes:

- “Tuning Command Line Tool Memory Settings” on page 119
- “QuickStart Command Tools” on page 120
- “Build Tools” on page 121

**See also**

- For a description of administrative commands provided with PolicyCenter, see “PolicyCenter Administrative Tools” on page 173 in the *System Administration Guide*.

## Tuning Command Line Tool Memory Settings

The `PolicyCenter/modules/configuration/etc/memory.properties` file specifies memory settings for the QuickStart application server and other gwpc tools. To change one or more of these settings edit the `xms`, `mxm`, and `maxperm` values for the class that runs the tool. Class information for commonly adjusted tools is provided with the command lists in “QuickStart Command Tools” on page 120 and “Build Tools” on page 121.

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

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

## QuickStart Command Tools

Use the following commands to control the QuickStart method of installing PolicyCenter on Windows only:

Command	Action
gwpc debug-start	Deprecated. Use dev-debug-shmem or dev-debug-socket instead.
gwpc dev-debug-shmem	Starts PolicyCenter in development mode using shared memory debugging.
gwpc dev-debug-socket	Starts PolicyCenter in development mode using socket debugging.
gwpc dev-deploy	Copies resources for the QuickStart application server.
gwpc dev-dropdb	<p>Prepares a new database for use by the Guidewire application. It will act upon the database configuration in <code>database-config.xml</code>, optionally selected by an <code>env</code> attribute by including the <code>-Denv="env"</code> parameter. If the database has any objects before the command is run, the objects are all dropped and not available for recovery except from a database backup. So, use extreme caution when using this command. The action of the command can be parameterized by settings in the <code>&lt;reset-tools-params&gt;</code> element underneath the <code>&lt;dbcp-connection-pool&gt;</code> element of the <code>&lt;database&gt;</code> element.</p> <p>For SQL Server, the command drops and creates a new database. It creates any filegroups named within the <code>&lt;database&gt;</code> element. The database files will be physically located where the server would put them by default. If specified, the <code>collate</code> attribute specifies the database collation. Otherwise, the default collation of the database server will be used. The command <code>SET READ_COMMITTED_SNAPSHOT ON WITH ROLLBACK IMMEDIATE</code> is issued on the new database. Since a <code>CREATE DATABASE</code> command is used, attributes of the model system database of the server are inherited.</p> <p>For SQL Server, the <code>dev-dropdb</code> command will create any file groups listed in the <code>upgrade</code> element of the database configuration. This does not happen on Oracle. However, the <code>dev-dropdb</code> command will not drop a database with file groups. If you have file groups configured, first drop the database using the Management Studio. Then you can run <code>dev-dropdb</code> to create the database with file groups.</p> <p>For Oracle, a Guidewire database corresponds to an Oracle schema: a user and all the objects owned by that user. So, for Oracle, this command drops all objects owned by the schema owner, and only the user is left with the required permissions. The <code>dev-dropdb</code> command requires SQLPlus, available with the Oracle client installation. Include <code>ORACLE_HOME\bin</code> in the <code>PATH</code>, so that <code>dev-dropdb</code> can locate SQLPlus.</p> <p>For Oracle, the <code>system-username</code> and <code>system-password</code> attributes of the <code>&lt;reset-tools-params&gt;</code> element must be specified so that the required permissions are available for these actions.</p> <p>Class: <code>com.guidewire.testharness.db.DBResetTool</code></p>
gwpc dev-start	<p>Starts the bundled QuickStart application server.</p> <p>Class: <code>com.guidewire.commons.jetty.GWServerJettyServerMain</code></p>
gwpc dev-stop	<p>Stops the bundled QuickStart application server.</p> <p>Class: <code>com.guidewire.commons.jetty.GWServerJettyServerStopMain</code></p>
gwpc dev-suspend-shmem	Starts PolicyCenter in development mode using shared memory debugging. Starts suspended.
gwpc dev-suspend-socket	Start PolicyCenter in development mode using socket debugging. Starts suspended.



## Build Tools

Guidewire supports all the build tools that it provides on Windows. Guidewire supports invoking a few build scripts on a non-Development Unix system by invoking Ant directly.

This topic contains:

- “Build Tools Supported on Windows” on page 122
- “Build Scripts Supported on Unix” on page 127

## Build Tools Supported on Windows

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

Command	Action
<code>gwpc -p</code>	Displays all gwpc command options.
<code>gwpc build-jboss-war-dbc</code>	<p>Builds the generic WAR file for JBoss including JDBC drivers. Use <code>gwpc build-jboss-war-dbc</code> if you are going to have PolicyCenter manage the database connection pool.</p> <p>You can include the Boolean parameter <code>config.war.dictionary=true</code> to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command:</p> <pre>gwpc build-jboss-war-dbc -Dconfig.war.dictionary=true</pre> <p>When <code>config.war.dictionary=true</code>, the command creates a dictionary folder within the WAR file. The dictionary folder contains data and security folders. These folders contain the <i>Data Dictionary</i> and <i>Security Dictionary</i> respectively. To view a dictionary, open <code>index.html</code> in the data or security folder.</p>
<code>gwpc build-jboss-war-jndi</code>	<p>Builds the generic WAR file for JBoss without JDBC drivers. Use <code>gwpc build-jboss-war-jndi</code> only if you are going to use a JNDI database connection managed by JBoss. See “Using a JNDI Data Source” on page 72.</p> <p>You can include the Boolean parameter <code>config.war.dictionary=true</code> to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command:</p> <pre>gwpc build-jboss-war-jndi -Dconfig.war.dictionary=true</pre> <p>When <code>config.war.dictionary=true</code>, the command creates a dictionary folder within the WAR file. The dictionary folder contains data and security folders. These folders contain the <i>Data Dictionary</i> and <i>Security Dictionary</i> respectively. To view a dictionary, open <code>index.html</code> in the data or security folder.</p>
<code>gwpc build-tomcat-war-dbc</code>	<p>Builds the generic WAR file for Tomcat including JDBC drivers. Use <code>gwpc build-tomcat-war-dbc</code> if you are going to have PolicyCenter manage the database connection pool.</p> <p>You can include the Boolean parameter <code>config.war.dictionary=true</code> to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command:</p> <pre>gwpc build-tomcat-war-dbc -Dconfig.war.dictionary=true</pre> <p>When <code>config.war.dictionary=true</code>, the command creates a dictionary folder within the WAR file. The dictionary folder contains data and security folders. These folders contain the <i>Data Dictionary</i> and <i>Security Dictionary</i> respectively. To view a dictionary, open <code>index.html</code> in the data or security folder.</p>
<code>gwpc build-tomcat-war-jndi</code>	<p>Builds the generic WAR file for Tomcat without JDBC drivers. Use <code>gwpc build-tomcat-war-jndi</code> only if you are going to use a JNDI database connection managed by JBoss. See “Using a JNDI Data Source” on page 72.</p> <p>You can include the Boolean parameter <code>config.war.dictionary=true</code> to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command:</p> <pre>gwpc build-tomcat-war-jndi -Dconfig.war.dictionary=true</pre> <p>When <code>config.war.dictionary=true</code>, the command creates a dictionary folder within the WAR file. The dictionary folder contains data and security folders. These folders contain the <i>Data Dictionary</i> and <i>Security Dictionary</i> respectively. To view a dictionary, open <code>index.html</code> in the data or security folder.</p>

Command	Action
<code>gwpc build-weblogic-ear-dbc</code>	Builds the EAR file for WebLogic including JDBC drivers. Use <code>gwpc build-websphere-ear-dbc</code> if you are going to have PolicyCenter manage the database connection pool.
<code>gwpc build-weblogic-ear-jndi</code>	Builds the EAR file for WebLogic without JDBC drivers. Use <code>gwpc build-websphere-ear-jndi</code> only if you are going to use a JNDI database connection managed by WebLogic. See “Using a JNDI Data Source” on page 72.
<code>gwpc build-websphere-ear-dbc</code>	Builds the EAR file for WebSphere including JDBC drivers. Use <code>gwpc build-websphere-ear-dbc</code> if you are going to have PolicyCenter manage the database connection pool.
<code>gwpc build-websphere-ear-jndi</code>	Builds the EAR file for WebSphere without JDBC drivers. Use <code>gwpc build-websphere-ear-jndi</code> only if you are going to use a JNDI database connection managed by WebSphere. See “Using a JNDI Data Source” on page 72.
<code>gwpc displaykey-diff</code>	<p>A display key difference tool that does the following:</p> <ul style="list-style-type: none"> <li>Compares each locale configured on the server against the master display key list.</li> <li>Generates a file that contains a list of any missing keys.</li> </ul> <p>See “Localizing Typecodes” on page 48 in the <i>Globalization Guide</i>.</p>
<code>gwpc export-l10ns</code> <code>-Dexport.file="translation_file"</code> <code>-Dexport.locale="language to export"</code>	<p>Exports a translation file from PolicyCenter into a file.</p> <p>The <code>-Dexport.file</code> parameter specifies the destination file.</p> <ul style="list-style-type: none"> <li>If you leave the import translation file in the same location, then enter only the name of the file to import.</li> <li>If you move the translation file to a different location, then enter an absolute path or a relative path to the file from the root of the installation directory.</li> </ul> <p>The <code>-Dexport.locale</code> parameter specifies the destination language to export. The <code>-Dexport.locale</code> parameter must match a PolicyCenter LanguageType typecode, such as <code>fr</code> or <code>ja</code>.</p> <p>See “Localizing Typecodes” on page 48 in the <i>Globalization Guide</i>.</p>
<code>gwpc import-l10ns</code> <code>-Dimport.file="translation_file"</code> <code>-Dimport.locale=destination_locale</code>	<p>Imports a translation file into the configuration.</p> <p>The <code>-Dimport.file</code> parameter specifies the file that contains the translations. It must be in the same format as an export file from Studio.</p> <p>The <code>-Dimport.locale</code> parameter specifies the destination language for the translations. The language must match a PolicyCenter LanguageType typecode, such as <code>fr</code> or <code>ja</code>.</p> <p>See “Localizing Typecodes” on page 48 in the <i>Globalization Guide</i>.</p>
<code>gwpc install-localized-module</code> <code>-Dmodule.file=ZipFileName</code> <code>-Dinstall.type={install upgrade}</code>	Installs or upgrades a language module. See “Installing Display Languages” on page 23 in the <i>Globalization Guide</i> .
<code>gwpc iterator-upgrade</code>	Upgrades all iterators on toolbar buttons and filters. This command is only used during upgrade from a prior major version. See “Running PCF Iterator Upgrade” on page 270 in the <i>Upgrade Guide</i> .
<code>gwpc regen-datamapping-split</code>	<p>Builds the data mapping files with files split out by table and typelist. Data mapping files represent fields present in the physical database. None of the virtual fields are represented.</p> <p>Class: <code>com.guidewire.tools.datamapping.DataMappingTool</code></p>
<code>gwpc regen-datamapping-together</code>	<p>Builds the data mapping files with all tables and typelists concatenated. represent fields present in the physical database. None of the virtual fields are represented.</p> <p>Class: <code>com.guidewire.tools.datamapping.DataMappingTool</code></p>

Command	Action
<pre> gwpc regen-dictionary -DmaxSPVInclusions=n -DoutputFormat={html xml} </pre>	<p>Generates the <i>Data Dictionary</i> and <i>Security Dictionary</i>. The <i>Data Dictionary</i> includes physical fields in the database and virtual fields in the data model. The <i>Security Dictionary</i> includes application permission keys, system permissions, and roles.</p> <p>Generate the dictionaries the first time you unzip PolicyCenter and each time you update the data model. Run the <code>gwpc regen-java-api</code> and <code>gwpc regen-soap-api</code> commands each time just prior to regenerating the security and data dictionaries.</p> <p><code>gwpc regen-dictionary</code></p> <p>To view either dictionary in HTML format, open the index file for it in a browser:</p> <pre> PolicyCenter/dictionary/data/index.html PolicyCenter/dictionary/security/index.html </pre> <p>You can generate the Data Dictionary and Security Dictionary in XML format, with associated XSD files. Use the generated XML and XSD files to import the <i>Data Dictionary</i> and <i>Security Dictionary</i> into third-party database design tools.</p> <pre> gwpc regen-dictionary -DoutputFormat=xml </pre> <p>This command generates the following XML and XSD files for the dictionaries:</p> <pre> GenericCenter/build/dictionary/data/entityModel.xml GenericCenter/build/dictionary/data/entityModel.xsd GenericCenter/build/dictionary/security/ securityDictionary.xml GenericCenter/build/dictionary/security/ securityDictionary.xsd </pre> <p>You can generate the dictionaries in HTML format while building a WAR file. See the description for <code>gwpc build-jboss-war</code> or <code>gwpc build-tomcat-war</code> for instructions.</p> <p>For more information, see “Regenerating the Data Dictionary and Security Dictionary” on page 32 in the <i>Configuration Guide</i>.</p> <p>This command performs PCF validation to catch errors in PCF files, such as:</p> <ul style="list-style-type: none"> <li>• Invalid expressions</li> <li>• Attributes that have no meaning when another attribute is set</li> <li>• Editable cells within non-editable objects</li> <li>• Illegal use of check boxes</li> <li>• Invalid arguments</li> <li>• Other errors</li> </ul> <p>Server commands that perform PCF validation, including <code>regen-dictionary</code>, <code>regen-pcfmapping</code>, and <code>verify-resources</code> can take a very long time to complete. The server performs a second-pass verification which, among other operations, verifies all possible combinations of modal sections on each PCF page. For example, a page with 12 modes that is used four times causes the server command to validate <math>12 \times 12 \times 12 \times 12 = 20,736</math> combinations. To limit the number of shared section verifications performed by these commands, specify the optional parameter <code>maxSPVInclusions</code>. This parameter defines the depth for second pass verification that limits the number of shared sections that are included in the verification of PCF types. For instance:</p> <pre> gwpc regen-dictionary -DmaxSPVInclusions=1000 </pre> <p>In this case, the second pass compilation of PCF files would stop after 1000 permutations of modal PCF files.</p> <p>Experiment with values for <code>maxSPVInclusions</code> between 1000 and 1000000 to achieve improved command completion times. However, be aware that limiting the validation depth means that some combinations of PCF modes and uses are not validated.</p> <p>The <code>maxSPVInclusions</code> property can only be specified as a positive integer value.</p> <p>For more information on second pass verification, see “Setting Verification Options” on page 116 in the <i>Configuration Guide</i>.</p> <p>Classes:</p> <pre> com.guidewire.tools.dictionary.data.DataDictionaryTool com.guidewire.tools.dictionary.security.SecurityDictionaryTool </pre>

Command	Action
<code>gwpc regen-from-wsc</code>	<p>Downloads the WSDL and XSD files for all WSC (Web Service Collection) files. Web service collection files encapsulate the set of resources necessary to connect to a web service on an external system. If you view a web service collection in Studio and click the <b>Fetch Updates</b> button, Studio gets the latest WSDL and XSD files from servers that publish those web services. This tool is equivalent to the <b>Fetch Updates</b> process, but runs from the command line and operates on all web services rather than one.</p> <p>See “Loading WSDL Locally Using Studio Web Service Collections” on page 76 in the <i>Integration Guide</i>.</p>
<code>gwpc regen-gosudoc</code>	<p>Generates Gosu API reference of the APIs available from Gosu within Studio. This command produces documentation at <code>PolicyCenter/build/gosudoc/index.html</code>. See “Gosu Generated Documentation (Gosudoc)” on page 38 in the <i>Gosu Reference Guide</i>.</p> <p>Class: <code>com.guidewire.tools.gosudoc.GosuDocMain</code></p>
<code>gwpc regen-java-api</code>	<p>Builds the Java API libraries to the <code>PolicyCenter/java-api</code> directory. See “Regenerating Integration Libraries and WSDL” on page 31 in the <i>Integration Guide</i>.</p>
<code>gwpc regen-pcfmapping -DmaxSPVInclusions=n</code>	<p>Builds the PCF mappings.</p> <p>This command performs PCF validation to catch errors in PCF files, such as:</p> <ul style="list-style-type: none"> <li>• Invalid expressions</li> <li>• Attributes that have no meaning when another attribute is set</li> <li>• Editable cells within non-editable objects</li> <li>• Illegal use of check boxes</li> <li>• Invalid arguments</li> <li>• Other errors</li> </ul> <p>Server commands that perform PCF validation, including <code>regen-dictionary</code>, <code>regen-pcfmapping</code>, and <code>verify-resources</code> can take a very long time to complete. The server performs a second-pass verification which, among other operations, verifies all possible combinations of modal sections on each PCF page. For example, a page with 12 modes that is used four times causes the server command to validate <math>12 \times 12 \times 12 \times 12 = 20,736</math> combinations. To limit the number of shared section verifications performed by these commands, specify the optional parameter <code>maxSPVInclusions</code>. This parameter defines the depth for second pass verification that limits the number of shared sections that are included in the verification of PCF types. For instance:</p> <pre>gwpc regen-pcfmapping -DmaxSPVInclusions=1000</pre> <p>In this case, the second pass compilation of PCF files would stop after 1000 permutations of modal PCF files.</p> <p>Experiment with values for <code>maxSPVInclusions</code> between 1000 and 1000000 to achieve improved command completion times. However, be aware that limiting the validation depth means that some combinations of PCF modes and uses are not validated.</p> <p>The <code>maxSPVInclusions</code> property can only be specified as a positive integer value.</p> <p>For more information on second pass verification, see “Setting Verification Options” on page 116 in the <i>Configuration Guide</i>.</p> <p>Class: <code>com.guidewire.tools.pcfmapping.PCFMappingWriterMain</code></p>
<code>gwpc regen-phone-metadata</code>	<p>Regenerates phone metadata in <code>config/phone/data</code>. Run this command if you have modified the phone metadata XML files</p>
<code>gwpc regen-rulereport</code>	<p>Generates an XML report describing the existing business rules. See “Generating a Rule Repository Report” on page 49 in the <i>Rules Guide</i>.</p>

Command	Action
gwpc regen-soap-api	Builds the web services (SOAP) API WSDL the PolicyCenter/soap-api directory. See “Regenerating Integration Libraries and WSDL” on page 31 in the <i>Integration Guide</i> .  Classes: com.guidewire.tools.wsdl.WSDLGenerator com.guidewire.util.webservices.axis.WSDLToJavaGenerator
gwpc regen-wsi-local	Regenerates WSDL for use in testing for all local web services. PolicyCenter generates the WSDL in the PolicyCenter\modules\configuration\gsrc\wsi\local directory. See “Generating WSDL On Disk” on page 54 in the <i>Integration Guide</i> .
gwpc regen-xsd	Builds the XSD files for data import. See “Constructing an XML File for Import” on page 101 in the <i>System Administration Guide</i> and “Importing Administrative Data” on page 95 in the <i>Integration Guide</i> .
gwpc studio	Runs Guidewire Studio.  Class: com.guidewire.studio.main.Main
gwpc upgrade-productmodel-to-structure -DtargetPath=target_path	Upgrades a 7.0.x or 8.0.x product model structure to the current 8.0.x structure. Use this command to upgrade the product model directory specified by targetPath to the current 8.0.x product model hierarchy.  Before running this command, Guidewire recommends that you make a backup copy of the specified product model directory as the command destructively modifies the target directory.  The targetPath argument is case-sensitive.
gwpc verify-checksum	Verifies module checksums.  Class: com.guidewire.tools.checksum.ModulesChecksumTool

Command	Action
<code>gwpc verify-resources -DmaxSPVInclusions=<i>n</i></code>	<p>Checks PCF files, XML schemas, and type loaders for errors.</p> <p>This command performs PCF validation to catch errors in PCF files, such as:</p> <ul style="list-style-type: none"> <li>• Invalid expressions</li> <li>• Attributes that have no meaning when another attribute is set</li> <li>• Editable cells within non-editable objects</li> <li>• Illegal use of check boxes</li> <li>• Invalid arguments</li> <li>• Other errors</li> </ul> <p>The command parses and compiles XML schemas and reports any errors.</p> <p>For each type loader, the command verifies the types supported by the loader. The command checks if a Gosu class compiles, or if a typelist can be retrieved properly.</p> <p>Server commands that perform PCF validation, including <code>regen-dictionary</code>, <code>regen-pcfmapping</code>, and <code>verify-resources</code> can take a very long time to complete. The server performs a second-pass verification which, among other operations, verifies all possible combinations of modal sections on each PCF page. For example, a page with 12 modes that is used four times causes the server command to validate <math>12 \times 12 \times 12 \times 12 = 20,736</math> combinations. To limit the number of shared section verifications performed by these commands, specify the optional parameter <code>maxSPVInclusions</code>. This parameter defines the depth for second pass verification that limits the number of shared sections that are included in the verification of PCF types. For instance:</p> <pre>gwpc verify-resources -DmaxSPVInclusions=1000</pre> <p>In this case, the second pass compilation of PCF files would stop after 1000 permutations of modal PCF files.</p> <p>Experiment with values for <code>maxSPVInclusions</code> between 1000 and 1000000 to achieve improved command completion times. However, be aware that limiting the validation depth means that some combinations of PCF modes and uses are not validated.</p> <p>For more information on second pass verification, see “Setting Verification Options” on page 116 in the <i>Configuration Guide</i>.</p> <p>The <code>maxSPVInclusions</code> property can only be specified as a positive integer value.</p>
<code>gwpc version</code>	Displays the product version.
<code>gwpc zip-changed-config -DoutputFile <i>filename.zip</i> [-DappRootDirectory <i>PolicyCenter Home</i>] [-DexcludeDir <i>directory1;directory2</i>]</code>	<p>Creates a ZIP file containing all files that are changed from the base configuration. Specify the output filename with the <code>-DoutputFile</code> parameter. You can also specify an application root directory by setting the <code>-DappRootDirectory</code> parameter. If you do not set <code>-DappRootDirectory</code>, the tool uses the directory above the <code>bin</code> directory as the root. The tool saves the output file relative to the application root. This file must not already exist.</p>

## Build Scripts Supported on Unix

On Windows, you can run various build scripts through the `gwpc` command. In contrast on Unix, you must invoke Ant directly with the following command:

```
ant -f PolicyCenter/modules/ant/build.xml buildScript
```

Substitute *buildScript* with one of the following values:

- `build-jboss-war-dbc` – Builds a generic WAR file for use with JBoss, including JDBC drivers
- `build-jboss-war-jndi` – Builds a generic WAR file for use with JBoss, without JDBC drivers

- `build-tomcat-war-dbc` – Builds a generic WAR file for use with Tomcat, including JDBC drivers
- `build-tomcat-war-jndi` – Builds a generic WAR file for use with Tomcat, without JDBC drivers
- `build-weblogic-ear-dbc` – Builds an EAR file for use with WebLogic, including JDBC drivers
- `build-weblogic-ear-jndi` – Builds an EAR file for use with WebLogic, without JDBC drivers
- `build-websphere-ear-dbc` – Builds an EAR file for use with WebSphere, including JDBC drivers
- `build-websphere-ear-jndi` – Builds an EAR file for use with WebSphere, without JDBC drivers

Although you can run many build scripts through the `gwpc` command on Windows, Guidewire supports running only the build scripts in the previous list by invoking Ant on Unix.