



Guidewire PolicyCenter™

Installation Guide

Release 10.1.2

© 2021 Guidewire Software, Inc.

For information about Guidewire trademarks, visit <http://guidewire.com/legal-notices>.

Guidewire Proprietary & Confidential — DO NOT DISTRIBUTE

Product Name: Guidewire PolicyCenter

Product Release: 10.1.2

Document Name: Installation Guide

Document Revision: 14-June-2021

Contents

Guidewire Documentation	8
About PolicyCenter documentation	8
Support	10
1 Introduction to installation	11
Installation topics roadmap	11
About installation roles	11
Servers, databases, and installation scenarios	12
Graphical overview of development and production environments	13
2 Preparing a PolicyCenter environment	15
Installation environments overview	15
About installing PolicyCenter in a development environment	15
About installing PolicyCenter in a production environment	16
Configuring the application server for Guidewire applications	16
Running PolicyCenter with the appropriate permissions	16
Supported uses of the application servers	16
General guidelines for application server installation	17
Fontconfig is required for Linux installations	17
Special considerations for clustered application server environments	17
Guidewire parameters for a clustered environment	17
Disabling IPv6 in a clustered environment	18
Configuring the JVM for environments without graphics	18
JVM heap size considerations	18
Heap size and memory in 32-bit and 64-bit applications	18
Operating system limits on heap size	19
Load balancers and Guidewire applications	19
Optional components installed with Guidewire applications	20
Configuring JBoss for Guidewire applications	20
Remove Stax2 JBoss JAR	20
Install a JBoss instance for free-text search	20
Configuring Tomcat for Guidewire applications	21
Hide Apache version information on error pages	21
Remove Tomcat examples directory	21
Do not implement the Tomcat native library	21
Increase the maximum concurrent threads	22
About session persistence on Tomcat	22
Install Tomcat as a Windows service	22
About heap size in Tomcat on Windows	23
Configuring WebLogic for Guidewire applications	24
Increase heap size for WebLogic	24
Configuring WebSphere for Guidewire applications	24
Increase heap size for WebSphere	24
Adjust ping parameters for WebSphere with Oracle	25
Verify WebSphere custom property	25
WebSphere database cluster management	25
WebSphere network deployment	25
Installing a WebSphere instance for free-text search	26

Configuring the database server for Guidewire applications	27
Overview of database configuration for Guidewire applications	27
Guidewire database direct update policy	27
Overview of database permissions used for Guidewire applications	28
About linguistic search collation	28
Configuring database compression	29
Configuring compression for Oracle	29
Configuring compression for SQL Server	31
Guidelines for configuring Oracle for PolicyCenter	34
Prerequisites to installing PolicyCenter on Oracle	34
Prepare an Oracle database for PolicyCenter	35
Database statistics generation for Oracle databases	36
About Oracle resource consumer groups	37
About Oracle SecureFile LOBs	37
About table partitioning for Oracle	38
About index partitioning for Oracle	39
About Oracle date interval partitioning	40
About Oracle on Amazon RDS	41
Guidelines for configuring SQL Server for PolicyCenter	41
Prerequisites to installing PolicyCenter on SQL Server	41
Install PolicyCenter on SQL Server	42
Configure SQL Server in Management Studio	42
Create a PolicyCenter database in SQL Server	42
About filegroups in SQL Server	44
Create filegroups in SQL Server	44
Grant Guidewire Data Management view permissions to pcUser	45
Disable the SQL Server autogrowth feature	45
Set the READ_COMMITTED_SNAPSHOT option	46
Configuring max degree of parallelism (MAXDOP) in SQL Server	46
About index partitioning for SQL Server	46
Final SQL Server tests	47
Development workstation requirements	47
Web client information	47
Levels of Guidewire support for web browsers	47
Enable DOM storage in Internet Explorer	48
Installing Java	48
Supported Java version	48
The Dynamic Code Evolution Virtual Machine	48
Install the DCEVM	49
Setting environment variables	50
Check environment variables	51
Documenting your environment	51
3 Installing a PolicyCenter development environment	53
Overview of development environment options	53
Using multiple development instances	53
Installing the QuickStart development environment	54
Advantages to using the QuickStart software	54
QuickStart development environment prerequisites	54
About the QuickStart application server	54
Configuring QuickStart ports	55
About the QuickStart default database	55
Install the bundled PolicyCenter QuickStart application server	55
QuickStart commands	56
Troubleshooting the QuickStart application server	56

Using the QuickStart database	57
H2 database configuration	57
Modify the QuickStart database file location.	58
Using SQL Server or Oracle in a development environment	58
Archiving in a development environment	58
Archive-related configuration files.	58
Enable archiving in Guidewire PolicyCenter.	59
Disabling Guidewire PolicyCenter archiving.	60
Install sample data	61
Using Gosu to configure sample data.	62
Verify sample data changes in PolicyCenter	62
4 Installing a PolicyCenter production environment.	63
Installing Guidewire PolicyCenter	63
Unpack the PolicyCenter configuration files	63
Configuring a database connection	64
About the database element	64
Improving screen performance in PolicyCenter	65
Mapping logical tablespaces to physical tablespaces	66
Configuration options for Oracle tablespaces	66
Configuration options for SQL Server filegroups	66
Configuration options for individual database tables	67
Configuration options for database table groups	68
Specifying additional database parameters.	68
The JDBC URL format	68
JDBC URL format for Oracle	68
JDBC URL format for SQL Server.	69
Obfuscating the database password	69
Use a password file to obfuscate the database password	70
About SQL Server JDBC logging	70
Using a JNDI data source	71
Configure PolicyCenter to use a direct JNDI data source.	71
Configure PolicyCenter to use an indirect JNDI data source	71
Create an Oracle JNDI data source on JBoss	72
Create a SQL Server JNDI data source on JBoss	73
Create an Oracle JNDI data source on Tomcat	73
Create a SQL Server JNDI data source on Tomcat	74
Create an Oracle JNDI data source on WebLogic	75
Create a SQL Server JNDI data source on WebLogic	76
Create an Oracle JNDI data source on WebSphere	77
Create a SQL Server JNDI data source on WebSphere	79
Deploying to the application server.	83
Installing PolicyCenter on JBoss in a production environment	83
Set Java options for JBoss	83
Add a servlet definition for JBoss.	83
Generate the PolicyCenter WAR file for JBoss.	84
Installing PolicyCenter on Tomcat in a production environment	84
Add servlet definitions for Tomcat	85
Generate and deploy the PolicyCenter WAR file for Tomcat	85
Installing PolicyCenter on WebLogic in a production environment	86
Add a servlet definition for WebLogic.	86
Enable HTTP authentication on WebLogic	86
Generate the PolicyCenter EAR file for WebLogic	86
Install the PolicyCenter EAR file on WebLogic	87
Ways to start PolicyCenter on WebLogic.	87

Installing PolicyCenter on WebSphere in a production environment	88
Add a welcome-file-list element to file web.xml.	88
Generate the PolicyCenter EAR file for WebSphere	89
Install the PolicyCenter EAR file on WebSphere	89
5 Additional PolicyCenter setup tasks	91
Additional installation information	91
Change the Super User (su) password	91
Generate Java API libraries	92
Integrating ClaimCenter and PolicyCenter.	92
Configure ClaimCenter to retrieve policy information	92
Configuring ClaimCenter to convert PolicyCenter objects	93
Configure PolicyCenter to retrieve claim information	94
Defining authentication between PolicyCenter and ClaimCenter	95
Configure ClaimCenter to send large loss notification to PolicyCenter	95
Test ClaimCenter large loss notification integration with PolicyCenter.	97
Integrating BillingCenter and PolicyCenter	97
Enable the policy system plugin in BillingCenter.	98
Enable the billing summary plugin in PolicyCenter	99
Defining authentication between PolicyCenter and BillingCenter	101
Start BillingCenter in an integrated environment	101
Start PolicyCenter in an integrated environment	102
Verify the integration between PolicyCenter and BillingCenter	103
Archiving in a production environment	104
About PolicyCenter Rating Management.	104
Disable reinsurance ceding work queue	104
Connecting a web client to PolicyCenter	105
Configure Microsoft Windows accessibility for Firefox	105
About single sign-on authentication	106
Configure single sign-on authentication	106
Starting PolicyCenter on the application server	109
Start PolicyCenter on JBoss	109
Start PolicyCenter on Tomcat on Windows.	109
Start Tomcat in same command prompt	109
Start PolicyCenter on WebLogic	110
Start PolicyCenter on WebSphere	110
Tune memory settings for application processes	110
6 Configuring free-text search	113
Overview of free-text search setup	113
Free-text search options for production and development	113
Simplified free-text search setup with embedded operation.	114
Guidewire Solr extension	114
Guidewire Solr home directory	114
Free-text batch load command	114
Securing database credentials for free-text batch load	115
Securing Guidewire Solr Extension	116
Requiring authentication in Guidewire Solr Extension	116
Configuring Guidewire Solr Extension to run with SSL activated	117
Configuring free-text search for embedded operation	117
Set up free-text search for embedded operation on QuickStart.	117
Set up free-text search for embedded operation on Tomcat	118
Configure free-text search for external operation	118
About the free-text batch load command	119
Configure the free-text batch load command for PolicyCenter.	121

7 Command reference	123
Core application tasks	123
Configuration upgrade tasks	125
Application server tasks	125
Globalization tasks	127
Integration tasks	127
Change verification tools	128
Documentation generation and other tools	128
Plugin development tasks	129
User interface tasks	130
JVM options for gwb build commands	130

Guidewire Documentation

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 the overall upgrade process, and describes how to upgrade your configuration and database. The intended readers are implementation engineers who must merge base application changes into existing application extensions and integrations.
<i>Configuration Upgrade Tools Guide</i>	Describes the tools and functionality provided by the Guidewire InsuranceSuite Configuration Upgrade Tools. The intended readers are implementation engineers who must merge base application changes into existing application extensions and integrations. Visit the Guidewire Community to access the <i>Configuration Upgrade Tools Guide</i> , which is available for download, separately from the main documentation set, with the Configuration Upgrade Tools.
<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 for PolicyCenter. The intended readers are all IT staff and configuration engineers.
<i>PCF Format Reference</i>	Describes PolicyCenter PCF widgets and attributes. The intended readers are configuration engineers. See the <i>Configuration Guide</i> .
<i>Data Dictionary</i>	Describes the PolicyCenter data model, including configuration extensions. The dictionary can be generated at any time to reflect the current PolicyCenter configuration. The intended readers are configuration engineers.
<i>Security Dictionary</i>	Describes all security permissions, roles, and the relationships among them. The dictionary can be generated at any time to reflect the current PolicyCenter configuration. The intended readers are 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 Guidewire Studio for PolicyCenter. The intended readers are business analysts who define business processes, as well as programmers who write business rules in Gosu.
<i>Guidewire 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

Document	Purpose
	readers are system architects and the integration programmers who write web services code or plugin code in Gosu or Java.
<i>REST API Client Guide</i>	Describes how to use the InsuranceSuite REST API Client to make outbound HTTP calls to internal or third-party REST services.
<i>Java API Reference</i>	Javadoc-style reference of PolicyCenter Java plugin interfaces, entity fields, and other utility classes. The intended readers are system architects and integration programmers.
<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>Gosu API Reference</i>	Javadoc-style reference of PolicyCenter Gosu classes and properties. The reference can be generated at any time to reflect the current PolicyCenter configuration. The intended readers are configuration engineers, system architects, and integration programmers.
<i>ISBTF and GUnit Testing Guide</i>	Describes the tools and functionality provided by InsuranceSuite for testing application behavior during an initial implementation or an upgrade. The guide covers functionality related to Behavior Testing Framework, GUnit, and Gosu functionality designed specifically for application testing. There are two sets of intended readers: business analysts who will assist in writing tests that describe the desired application behavior; and technical developers who will write implementation code that executes the tests.
<i>Glossary</i>	Defines industry terminology and technical terms in Guidewire documentation. The intended readers are everyone who works with Guidewire applications.
<i>Advanced Product Designer Guide</i>	Advanced Product Designer is a tool that helps you, a business user, design, simulate, and deploy an insurance product. The intended readers are business analysts who understand insurance products, business systems analysts who liaise between business analysts and IT, project managers, and IT who provides technical expertise in areas such as programming, testing, and databases.
<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.
<i>REST API Framework</i>	Describes the Guidewire InsuranceSuite framework that provides the means to define, implement, and publish REST API contracts. It also describes how the Guidewire REST framework interacts with JSON and Swagger objects. The intended readers are system architects and integration programmers who write web services code or plugin code in Gosu or Java.

Conventions in this document

Text style	Meaning	Examples
<i>italic</i>	Indicates a term that is being defined, added emphasis, and book titles. In monospace text, italics indicate a variable to be replaced.	A <i>destination</i> sends messages to an external system. Navigate to the PolicyCenter installation directory by running the following command: <pre>cd installDir</pre>
bold	Highlights important sections of code in examples.	<pre>for (i=0, i<someArray.length(), i++) { newArray[i] = someArray[i].getName() }</pre>
narrow bold	The name of a user interface element, such as a button name, a menu item name, or a tab name.	Click Submit .

Text style	Meaning	Examples
monospace	Code examples, computer output, class and method names, URLs, parameter names, string literals, and other objects that might appear in programming code.	The getName method of the IDoStuff API returns the name of the object.
<i>monospace italic</i>	Variable placeholder text within code examples, command examples, file paths, and URLs.	Run the startServer <i>server_name</i> command. Navigate to <code>http://server_name/index.html</code> .

Support

For assistance, visit the Guidewire Community.

Guidewire customers

<https://community.guidewire.com>

Guidewire partners

<https://partner.guidewire.com>

Introduction to installation

Installing PolicyCenter starts with understanding the installation options that are available and which option to choose.

Installation topics roadmap

The following topics describe how to install PolicyCenter.

Topic	Description
"About installation roles" on page 11	Overview that discusses the different ways to install PolicyCenter.
"Graphical overview of development and production environments" on page 13	Diagram that shows how the PolicyCenter development and production environment interact
"Preparing a PolicyCenter environment" on page 15	Steps to prepare a development or production environment for PolicyCenter.
"Installing a PolicyCenter development environment" on page 53	How to install a PolicyCenter development environment using the QuickStart server and database or Tomcat.
"Installing a PolicyCenter production environment" on page 63	How to deploy PolicyCenter to an application server and database server production environment.
"Additional PolicyCenter setup tasks" on page 91	Optional installation tasks you may want to perform after you complete the installation and deployment of a PolicyCenter development or production environment.
"Command reference" on page 123	Descriptions of the QuickStart and build commands. Topic the <i>System Administration Guide</i> describes PolicyCenter command utilities used by system administrators.

About installation roles

The people that install and develop Guidewire PolicyCenter have various roles. The following list describes these roles.

Role	Description
Demonstrator or Trainer	Starts up the application quickly, loads sample data and demonstrates features.
Application Developer	Changes the behavior of the application including the user interface, rules, and application logic.
Integration Developer	Develops software to connect PolicyCenter to external systems.
Conversion Developer	Performs analysis and mapping of legacy data structures to Guidewire application data model.
Build master deploying to testing and production	Deploys finished application to test and production environments.

Note: For information about supported servers and databases and the installation scenarios they support, see “Servers, databases, and installation scenarios” on page 12.

See also

- “Installation environments overview” on page 15
- “Installing a PolicyCenter development environment” on page 53
- “Installing a PolicyCenter production environment” on page 63

Servers, databases, and installation scenarios

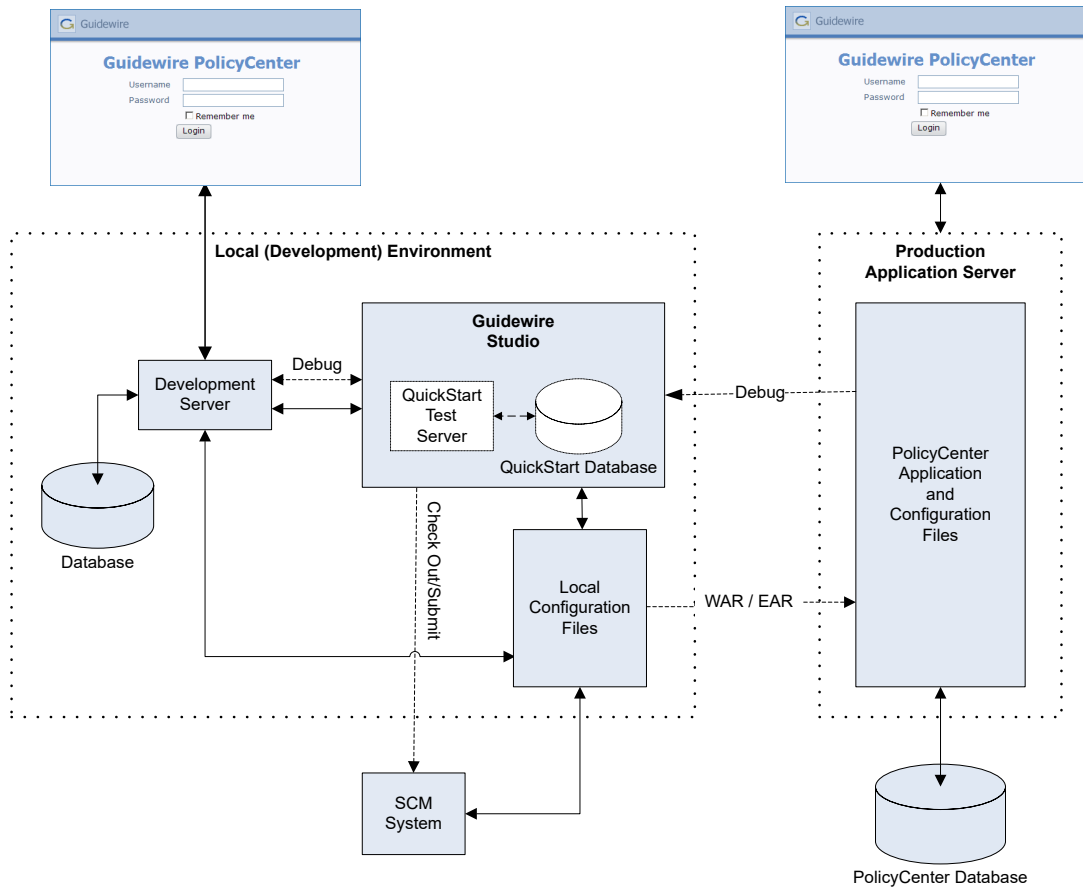
To help you choose an installation scenario, the following table describes the uses of servers and databases that you can install with PolicyCenter.

Install element	Type	Good to know	More information
QuickStart application server	Bundled	You can immediately use the QuickStart server without configuring it. It does not build a WAR or EAR file, which it is necessary to do before deployment with other servers. Use the QuickStart application server for demonstration or development (dev) environments. It is not possible to run PolicyCenter in production (prod) mode on the QuickStart server.	• “Advantages to using the QuickStart software” on page 54 • “Installing the QuickStart development environment” on page 54 • “About the QuickStart application server” on page 54 • <i>System Administration Guide</i>
JBoss application server	Optional	Suitable for production environments.	• “Configuring the application server for Guidewire applications” on page 16 • “Installing PolicyCenter on JBoss in a production environment” on page 83
Tomcat application server	Optional	Suitable for production environments.	• “Configuring the application server for Guidewire applications” on page 16 • “Installing PolicyCenter on Tomcat in a production environment” on page 84
WebSphere application server	Optional	Suitable for production environments.	• “Configuring the application server for Guidewire applications” on page 16 • “Installing PolicyCenter on WebSphere in a production environment” on page 88

Install element	Type	Good to know	More information
WebLogic application server	Optional	Suitable for production environments.	“Configuring the application server for Guidewire applications” on page 16 “Installing PolicyCenter on WebLogic in a production environment” on page 86
QuickStart database	Bundled	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. You can use the QuickStart database for demonstration or development environments. Guidewire does not support QuickStart for a production environment. Guidewire does not support upgrades to the QuickStart database. Configuring your application sometimes requires extending the data model, which might require dropping the database. With QuickStart, it is not possible to have more than one database connection at a time.	“Advantages to using the QuickStart software” on page 54 “Using the QuickStart database” on page 57
Oracle database	Optional	It is possible to use the Oracle database for both development and production.	“Configuring the database server for Guidewire applications” on page 27 “Guidelines for configuring Oracle for PolicyCenter” on page 34 “Configuring a database connection” on page 64
SQL Server database	Optional	It is possible to use the Microsoft SQL Server database for both development and production.	“Configuring the database server for Guidewire applications” on page 27 “Guidelines for configuring SQL Server for PolicyCenter” on page 41 “Configuring a database connection” on page 64

Graphical overview of 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

You must install and configure necessary system components so that your network can support PolicyCenter. Additionally, there are preparatory steps to follow to deploy a production instance of PolicyCenter.

IMPORTANT The versions of third-party products that Guidewire supports for this release are subject to change without notice. See the *Supported Software Components* knowledge article for current system and patch level requirements. Visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

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, ensure that you have the appropriate software versions to support PolicyCenter. Guidewire recommends strongly that you obtain support contracts with vendors for all tiers of your application infrastructure.

Although production environments can run on operating systems other than Microsoft Windows, all development environments must run on the Windows operating system. However, you can build PolicyCenter on a non-development Unix system prior to deploying the PolicyCenter application.

About installing PolicyCenter in a development environment

Guidewire supports the bundled QuickStart application server for development environments.

IMPORTANT For PolicyCenter development, all application builds must use the appropriate Oracle JDK. For current system and patch level requirements, visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

For all development environments, review the following topics:

- “Development workstation requirements” on page 47
- “Web client information” on page 47
- “Running PolicyCenter with the appropriate permissions” on page 16 (Windows information only)
- “Installing Java” on page 48
- “Setting environment variables” on page 50
- “Documenting your environment” on page 51

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

- “Configuring the database server for Guidewire applications” on page 27

After reviewing the relevant development environment information, proceed to “Installing a PolicyCenter development environment” on page 53.

See also

- “Development workstation requirements” on page 47

About installing PolicyCenter in a production environment

In production environments, Guidewire strongly recommends that you allocate separate dedicated hardware for the application server and database server tiers. Reserve this hardware solely for use with Guidewire 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 develop PolicyCenter on a Windows system prior to deploying PolicyCenter.

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

For production environments, review the following material:

- “Preparing a PolicyCenter environment” on page 15
- “Installing a PolicyCenter production environment” on page 63

Configuring the application server for Guidewire applications

The PolicyCenter application server relies on a third-party servlet container for execution and connection services. There are some adjustments you must make to the supported application servers.

Running PolicyCenter with the appropriate permissions

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:

Micro- soft Win- dows	If your network servers are Microsoft Windows systems, create a user with the Log on as a service right. Ensure that this user is not a member of any group. 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 “Install Tomcat as a Windows service” on page 22 for more information.
UNIX	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 that the PolicyCenter processes run in the correct mode, create a specific user account on each server and run the corresponding applications under these accounts.

Supported uses of the application servers

Guidewire supports the following servers for production or development environments:

Production envi- ronments	Guidewire supports JBoss, WebSphere, WebLogic, and Tomcat application servers.
Development envi- ronments	Guidewire provides the best support for the bundled QuickStart application server. Guidewire also supports JBoss, WebSphere, WebLogic, and Tomcat application servers for development use. However, because these application servers do not reload resources modified in Studio without a rebuild and redeploy, they are not ideal for development work.

See also

- For information about the specific application server versions Guidewire supports for PolicyCenter 10.1.2, visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.
- For information on configuring the QuickStart application server, see “About the QuickStart application server” on page 54.

General guidelines for application server installation

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

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

Use only the JDK or JRE specified in the *Supported Software Components* knowledge article, 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.

PolicyCenter does not support the security manager feature for any application server. Do not enable this feature.

See also

- For information about the specific JDK or JRE versions that Guidewire supports for PolicyCenter 10.1.2, visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

Fontconfig is required for Linux installations

Some versions of Linux do not include the fontconfig library. This library is required for PolicyCenter 10.1.2.

Verify that the fontconfig library is installed, and add it to your Linux system if the library is not already available.

Special considerations for clustered application server environments

There are special considerations for running Guidewire applications in clustered environments.

See also

- *System Administration Guide*

Guidewire parameters for a clustered environment

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

- `ClusteringEnabled`
- `ClusterMemberPurgeDaysOld`
- `ClusterMemberRecordUpdateIntervalSecs`
- `ClusterStatisticsMonitorIntervalMins`
- `ConfigVerificationEnabled`
- `PDFMergeHandlerLicenseKey`

See also

- See the *Configuration Guide* for parameter descriptions and suggested values.

Disabling IPv6 in a clustered environment

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
```

See the following table for details.

Tomcat	Add the specified option to the CATALINA_OPTS environment variable.
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 following line: <pre>JAVA_OPTIONS="{JAVA_OPTIONS} {JAVA_PROPERTIES} -Dwlw.iterativeDev={iterativeDevFlag} -Dwlw.testConsole={testConsoleFlag} -Dwlw.logErrorsToConsole={logErrorsToConsoleFlag}"</pre>
WebSphere	Add the option by using the Administrative Console: - Navigate to Servers→Application servers→server. - In the Server Infrastructure section, navigate to the following location: Java and process management→Process definition→Java virtual machine - Click Custom Properties. - Add the following option: java.net.preferIPv4Stack=true

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

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 JVM parameter `java.awt.headless` 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.

See also

- “About heap size in Tomcat on Windows” on page 23
- “Increase heap size for WebLogic” on page 24
- “Increase heap size for WebSphere” on page 24

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

Heap size and memory in 32-bit and 64-bit applications

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

Note: The terms *32-bit* and *64-bit* refer to the size of the pointer used to reference an address.

Typically, a 64-bit JVM has an approximately 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 heap sizes of 1024MB for a 32-bit JVM and 2048 MB for a 64-bit JVM.

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

Operating system limits on heap size

The following tables group operating system heap size limitations by application server.

Note: The following tables that list limits on heap size serve only as a starting point for tuning optimal JVM settings for your configuration. They are useful if you are installing PolicyCenter and want to start development work. Since each deployment needs to be tuned for its dataset, the heap sizes depend on your usage and configuration. Production systems require careful sizing. Consult Guidewire Services for assistance.

Heap sizing for JBoss, WebLogic, and Tomcat

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

Heap sizing 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 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.

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.

Heap sizing for testing

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

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

Load balancers and Guidewire applications

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.

See also

- *System Administration Guide*

Optional components installed with Guidewire applications

If you plan to have PolicyCenter send email through business logic, have an SMTP-compatible email server available, such as Microsoft Exchange or UNIX Sendmail. 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 for Guidewire applications

JBoss requires special configurations to run it as the application server for PolicyCenter.

Remove Stax2 JBoss JAR

About this task

Remove the following JAR file from your JBoss installation, if present:

```
modules/system/layers/base/org/codehaus/woodstox/main/stax2-api-3.1.1-redhat-3.jar
```

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.

Install a JBoss instance for free-text search

About this task

Enabling 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 that is 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

-
- “Install a Tomcat instance for free-text search ” on page 23
- “Installing a WebSphere instance for free-text search” on page 26

Configuring Tomcat for Guidewire applications

This topic provides notes on installing Tomcat to run PolicyCenter. Log in as a system administrator to install Tomcat.

Hide Apache version information on error pages

About this task

By default, Apache HTTP server discloses the Apache version number in HTTP response headers and Apache error pages. This information can reveal valuable details to an attacker about possible vulnerabilities in the software. If you are using Apache HTTP Server, Guidewire recommends that you customize your installation to not return this information.

Procedure

1. Open the Apache HTTP server `httpd.conf` file, located in the `conf` directory.
2. Change `ServerSignature` to `Off`.
3. Change `ServerTokens` to `Prod`.
4. Save `httpd.conf`.
5. Restart the Apache HTTP Server.

Remove Tomcat examples directory

About this task

Guidewire recommends that you remove the following directory from any Guidewire production implementation that uses Tomcat:

```
TOMCAT_HOME/webapps/examples
```

There have been security vulnerabilities reported for some versions of the example scripts shipped with Tomcat.

Do not implement the 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 mixture of coding languages requires the use of the 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 calculation 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:
```

You can safely ignore this message.

Increase the maximum concurrent threads

About this task

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.

Procedure

1. In the Tomcat installation directory, find and open file `conf/server.xml`.
2. Find the definition for the http connector. It looks similar to the following:

```
<Connector port="8180" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443" />
```

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

```
<Connector port="8180" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443"
  maxThreads="200" />
```

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

About session persistence on Tomcat

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.

Whenever you restart the Tomcat server, it tries to restore the sessions. However, the session contents are meaningful only 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)
  ...
```

Guidewire recommends that you configure Tomcat to disable session persistence.

Disable session persistence on Tomcat

Procedure

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

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

3. Save `context.xml`.

Install Tomcat as a Windows service

About this task

If you plan on installing only one application that uses Tomcat, you can use 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.

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

Increase Tomcat service heap size on Windows

Procedure

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

Increase Tomcat heap size on Windows if Tomcat is not a service

Procedure

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.
For values to set in this step, see “JVM heap size considerations” on page 18.
7. Click **OK** until the properties settings dialog closes.

Install a Tomcat instance for free-text search

About this task

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

-
- “Install a JBoss instance for free-text search” on page 20
- “Installing a WebSphere instance for free-text search” on page 26

Configuring WebLogic for Guidewire applications

There are special considerations for WebLogic if you install it as the application server for a Guidewire application.

Increase heap size for WebLogic

About this task

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

Procedure

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.
3. (Optional) To run the server in production mode, remove the `-Dgw.server.mode` argument listed in “Increase heap size for WebLogic” on page 24.
 By default, PolicyCenter starts in production mode on all application servers except the bundled QuickStart server.

Configuring WebSphere for Guidewire applications

There are special considerations for WebSphere whenever you install it as the application server for a Guidewire application.

Increase heap size for WebSphere

About this task

To support Guidewire applications in WebSphere, increase minimum heap size to 256 and maximum heap size to 1024.

Procedure

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

Adjust ping parameters for WebSphere with Oracle

About this task

If WebSphere times out while communicating with Oracle, Guidewire recommends that you set the following values:

Server Parameter	Value
ping interval	2000
ping timeout	6000

Verify WebSphere custom property

About this task

PolicyCenter client-side JavaScript must be able to read the value of the `csrfToken` cookie.

Procedure

Verify that the WebSphere web container custom property `com.ibm.ws.webcontainer.HTTPOnlyCookies` is *not* configured to enable the HTTPOnly attribute on the PolicyCenter `csrfToken` cookie.

The `com.ibm.ws.webcontainer.HTTPOnlyCookies` contains a comma separated list of cookies that are given the HTTPOnly attribute. An asterisk value means that all cookies are given the HTTPOnly attribute.

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, for example:

Guidewire Cache Coherency for Clustering

PolicyCenter provides cache coherency to support clustering by maintaining an internal cache of objects for performance reasons. The Guidewire implementation of clustering for cache coherency is completely independent of the application server.

WebSphere ND Clustering

Clustering provided by WebSphere ND is different from PolicyCenter cache coherency. 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.

WebSphere ND provides the following features that you can 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 Guidewire Environment Specific Configuration settings.

Load balancing

WebSphere ND provides load balancing capabilities. See “Load balancers and Guidewire applications” on page 19.

See also

- *System Administration Guide*

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 that is 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.

See also

- “Change the port number in WebSphere for free-text search” on page 26

Change the port number in WebSphere for free-text search

Procedure

1. Start the PolicyCenter 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 of 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	Enter
Host Name	Asterisk (*)
Port	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.

Configuring the database server for Guidewire applications

Guidewire recommends that you implement the guidelines contained in this topic as you install and configure the database server.

Overview of database configuration for Guidewire applications

PolicyCenter 10.1.2 supports Oracle and SQL Server for production environments. See the *Supported Software Components* knowledge article for information about which specific database server versions Guidewire supports for PolicyCenter 10.1.2. Visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

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 and same physical location. The maximum time difference allowed between the application server and database server is 29 minutes. If you use database clustering, synchronize all nodes in the database cluster with each other.

Guidewire database direct update policy

PolicyCenter runs on a Relational Database Management System (RDBMS). You can use SQL or other query tools directly in a read-only manner to extract or view data. Such read-only queries, depending on their scope and how they are written, can negatively effect overall database performance even though they do not modify any data. Therefore, Guidewire recommends that you run SQL queries in a replica or copy of their production database, rather than the production database itself. For applications such as data warehouses and intensive reporting, Guidewire recommends that you explore mechanisms for replicating or summarizing data into a production reporting database for this purpose. This practice can help unexpected production performance issues due to intensive reporting requirements or lengthy queries.

Internal application logic, embedded in PolicyCenter application code and APIs, maintains a variety of data and metadata that relates to your application data. This might not be obvious from review of the RDBMS table structure. Examples include:

- Calculations of summary table data for reporting
- Caching of application data in memory for faster access
- Tracking of state information associated with the underlying data, such as the processing state of an integration message

For this reason, never use SQL to directly update the underlying RDBMS. Any such direct SQL updates can leave the data in an inconsistent state. Guidewire can require you to restore the database to a previous state if you require support after performing such an update query. Guidewire Support is not able to assist you with diagnosing and correcting application issues caused by your database queries. It is your responsibility to restore PolicyCenter to a consistent state.

WARNING Guidewire supports the built-in automatic database upgrade process only for Guidewire InsuranceSuite products. Guidewire explicitly does not support any alternative process that executes SQL DDL commands on the database.

If you have a legitimate need to update underlying application data, Guidewire recommends that you use Guidewire APIs, either Java or Gosu, to perform the necessary updates. This ensures that you do not miss any critical side effects of the updates in the process of altering the data. Using Guidewire APIs to update application data is safer than using SQL queries with regard to consistency. However, with any programming language or API it is still possible to update data incorrectly or in ways that do not perform well. Therefore, before using the APIs, Guidewire

strongly recommends that you review your intended updates with your Guidewire Support Partner and/or Guidewire Professional Services team.

In the rare case in which no API exists to correct a data corruption problem, Guidewire can advise you on the SQL queries to use to correct these problems. In these cases, the SQL queries used to update the database must be written, or approved, by Guidewire. This process ensures that all SQL queries use correct logic and that you take all potential side effects into account.

Do not apply any other SQL queries to modify data in a PolicyCenter database. Guidewire does not provide, nor review, such queries for situations in which an API or supported alternate method is available.

Overview of database permissions used for Guidewire applications

PolicyCenter must be able to connect to your database through a user. Create a user called pcUser in your database. PolicyCenter connects to the user named pcUser. The pcUser must have the correct permissions. The following tables lists the permissions required for each database:

Data-base	Permissions Required
Oracle	The pcUser must have the following permissions on the PolicyCenter Oracle database: • Alter session • Create operator • Create procedure • Create sequence • Create session • Create table • Create trigger • Create view • Query rewrite • Select any dictionary • Load Class (if using Amazon RDS and GWNormalize Java function to configure search rules.) If your users want to see statspack data on the Server Tools Info Pages→Oracle Statspack screen, grant the pcUser access to the PolicyCenter performance statistics (perfstat) tables.
SQL Server	The pcUser must have the public and db_owner roles on the PolicyCenter SQL Server database. 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 in PolicyCenter. The server login account must also have view database state permission on each PolicyCenter data management view. The data management views all start with a sys.dm_ prefix.

See also

- *System Administration Guide*

About linguistic search collation

It is possible to configure how PolicyCenter searches and sorts search results. For example, you can configure whether searching is accent-sensitive or accent-insensitive, or if it is case-sensitive or case-insensitive.

See also

- *Globalization Guide*

Configuring database compression

It is possible to configure compression for the databases that you use with your Guidewire applications. 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 of compression is that the database requires more CPU time to compress and decompress data. However, queries that require either a table scan or a full index scan require fewer physical and logical reads because more rows fit on a single data page.

Consult with Guidewire Support about whether database compression can improve the overall performance of your PolicyCenter implementation.

Note: 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 configure Oracle compression options for PolicyCenter by using the `<ora-db-ddl>` element of `database-config.xml`. Compression options can apply to the entire database, to specific tables, or to specific indexes of a table.

Note: The compression settings examples in the following topics merely demonstrate the syntax to use to configure compression. Do not use these examples as formal guidelines for compression settings for your environment.

Oracle database compression

You can specify Oracle database compression options for PolicyCenter by using the `<ora-compression>` element of `<ora-db-ddl>` in `database-config.xml`. The `<ora-compression>` element has the following syntax:

```
<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.

The values that you can set for these two attributes are described in the topics that follow.

See also

- *System Administration Guide*

Oracle table compression

You can override options for all tables in the database by setting the `table-compression` attribute of the `<ora-compression>` element to `NONE`, `BASIC` or `ADVANCED`. For general syntax, see the preceding topic.

You can override options for a specific table by adding an `<ora-table-compression>` element and setting the `table-compression` attribute to `NONE`, `BASIC` or `ADVANCED`. 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>
```

```

    </ora-db-ddl>
  </upgrade>
</database>

```

For the `ora-table-compression` `table-compression` attribute:

- 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.

Note: 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">
  ...
  <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>

```

See also

- *System Administration Guide*

Oracle index compression

You can set the `index-compression` attribute of the `<ora-compression>` and `<ora-index-ddl>` elements to `true` or `false`.

An `index-compression` value of `true` specifies:

- Compression for all columns but the last for unique indexes
- Compression for all columns for non-unique indexes

Depending on which element you set, you can override compression for all indexes in a database, for all indexes on a specific table, or for specific indexes.

- You can override compression options for all indexes in a database by specifying the `index-compression` attribute of the `<ora-compression>` element. For general syntax, see the example in the preceding topic, “Oracle database compression” on page 29.
- 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>

```

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.

The following example specifies the following overrides:

- 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="xx_Activity">
          <ora-index-ddl key-columns="PublicID,Retired" index-compression="false" />
        </ora-table-ddl>
        <ora-table-ddl table-name="xx_Workflow">
          <ora-table-compression index-compression="false" />
        </ora-table-ddl>
      </ora-db-ddl>
    </upgrade>
  </database>

```

IMPORTANT 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.

See also

- *System Administration Guide*

Configuring compression for SQL Server

You can configure Microsoft SQL Server compression options for PolicyCenter by using the `<mssql-db-ddl>` element of `database-config.xml`. Compression options can apply to the entire database, to specific tables, or to specific indexes of a table.

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

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

SQL Server database compression

You can specify SQL Server database compression options for PolicyCenter by using the `<mssql-compression>` element of `<mssql-db-ddl>` in `database-config.xml`:

```

<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.

Settings for `table-compression` and `index-compression` can apply to the entire database, a table, or an index, depending on the XML element to which the attribute is applied. In general, these values mean the following:

- 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. SQL Server applies page compression only if the page becomes full. For page compression, the following operations happen in the listed 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.

See also

- *System Administration Guide*

SQL Server table compression

You can override options for all tables in the database by setting the `table-compression` attribute of the `<mssql-compression>` element to `NONE`, `PAGE`, or `ROW`.

For syntax and a description of the options for `table-compression`, see the preceding topic.

You can override options for a specific table by adding an `<mssql-table-compression>` element and setting the `table-compression` attribute to `NONE`, `PAGE`, or `ROW`. 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>
```

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">
...
<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>
```

See also

- *System Administration Guide*

SQL Server index compression

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.

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 override compression options for all indexes in the database by setting the `index-compression` attribute on the `<mssql-compression>` element to `NONE`, `PAGE`, or `ROW`.

For this syntax and a general description of the options for `index-compression`, see “SQL Server database compression” on page 31.

- You can override compression options for all indexes on a specific table by setting 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.
 - 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.

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>
```

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>
```

See also

- *System Administration Guide*

Guidelines for configuring Oracle for PolicyCenter

Guidewire recommends that you implement the guidelines in this topic as you install and configure the Oracle database server to work with PolicyCenter.

Note: See the *Supported Software Components* knowledge article for information about which specific Oracle versions Guidewire supports for PolicyCenter 10.1.2. Visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

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)*. Note that PolicyCenter does not require Oracle Spatial, an extra-cost option that includes all Oracle Locator features plus advanced features for spatial data.

Prerequisites to installing PolicyCenter 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 document:

<http://www.oracle.com/technetwork/database/performance/opt-storage-conf-130048.pdf>

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

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

- Operating system
- Available memory
- Database release

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

The Oracle database supports server-side caching that can help increase 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 cannot 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 open source tool, IOzone (www.iozone.org).

Prepare an Oracle database for PolicyCenter

About this task

IMPORTANT

Guidewire supports only single-byte character sets that are a strict superset of ASCII, and AL32UTF8 or UTF8 for Unicode. Use only 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, configure the Oracle instance to set `nls_length_semantics` to `char`. Otherwise, the application server will not start.

Note: Oracle recommends that you do not set the `nls_length_semantics` parameter to `char` (due to concerns about third party applications relying on the value being set to `byte`), but Guidewire specifically relies on having it set to `char`. For more details, see *Oracle support MOS Doc ID 144808.1*.

Guidewire does not support non-ASCII characters in database schema object names and qualifiers. Without limitation, these names and qualifiers include table names and column names. If you do not restrict schema object names and qualifiers to ASCII characters, an exception will result.

Procedure

1. Create a new database instance for PolicyCenter.

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

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 “About Oracle SecureFile LOBs” on page 37.

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. (Optional) In a production environment, edit the PATH environment variable and move any Oracle directories to the end of the variable, after the PolicyCenter directories.

This action prevents potential conflicts with PolicyCenter files as Oracle adds directories to the PATH environment variable if you run the database and application servers on the same computer.

Note: 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 and verify that all the tablespaces are visible.
9. Define the method you will use to gather Oracle database statistics. Select and configure your choice as outlined in .

Database statistics generation for Oracle databases

There are several different ways in which to generate database statistics in Guidewire PolicyCenter if using an Oracle database:

- Use the Oracle Autotask infrastructure to manage the task of gathering database table statistics.
- Run PolicyCenter batch processing DBStats periodically to collect database table statics.

You enable the use of each method by setting the value of the useoraclestatspreferences attribute on the <tablestatistics> element in file database-config.xml.

use... attribute	Statistics	Description	More information
true	Oracle AutoTask	Set AUTOSTATS_TARGET parameter to AUTO.	
false (Default)	DBStats batch processing	Set AUTOSTATS_TARGET parameter to ORACLE.	

Note: A change in the value of useoraclestatspreferences takes effect only during an application upgrade.

Note: Due to Oracle Bug 27818902, using Oracle AutoTask to gather database statistics can create certain issues in Guidewire PolicyCenter. See the **Major issues and changes** section of the Release Notes for details.

Using DBStats batch processing

If using DBStats batch processing to manage the collection of database statistics:

- Do not execute Oracle dbms_stats manually.
- Manually execute, or schedule, DBStats batch processing.
- Set AUTOSTATS_TARGET parameter to ORACLE using the DBMS_STATS.SET_GLOBAL_PREFS procedure. This action ensures that the automated task 'auto optimizer stats collection' gathers statistics for the Oracle Dictionary only.

Guidewire recommendations for Oracle database installations

- Guidewire recommends that Oracle implementations only update database statistics during quiet periods, such as weekends or nights, so that these updates do not occur while PolicyCenter is under heavy load. By default, updating statistics on a table or index invalidates existing query plans related to that table or index.
- Guidewire recommends that Oracle implementations use the `NO_INVALIDATE => AUTO_INVALIDATE` option while updating database statistics. This is the default option. This option is also what the Guidewire Database Statistics batch process uses, unless the configuration parameter `DiscardQueryPlansDuringStatsUpdateBatch` is set to `true`.

Setting `NO_INVALIDATE => FALSE` to force immediate invalidation of query plans has a high likelihood of causing issues with concurrent batch updates. Using `AUTO_INVALIDATE` greatly reduces this risk. Ideally, set the `_optimizer_invalidation_period` parameter to a low value (a few minutes) to reduce the time window during which Oracle might invalidate a plan.

About Oracle resource consumer groups

Note: Use the information in this topic only if you 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 that it detects during server start up and reverts to that group following the policy query.

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

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

```
<database name="PolicyCenterDatabase" dbtype="oracle">
  ...
  <oracle-settings db-resource-mgr-cancel-sql="" />
  ...
</database>
```

- The Oracle resource manager plan is set at the system level.
- User `pcUser` has privileges to switch between the two resource groups.

PolicyCenter checks these conditions at server start up.

See also

- *System Administration Guide*

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

Use the following syntax in the `<database>` block to specify the use of compressed SecureFile LOBs for all LOBs in the database.

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

Use the following syntax in the <database> block to specify the use of compressed SecureFile LOBs for all LOBs on a particular table.

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

If you configure any LOBS to be SecureFile LOBs, and:

- You configure the LOB tablespace, you must manage the LOB tablespace with Automatic Segment Space Management.
- You do not configure the LOB tablespace, you must manage the ADMIN, OP and STAGING tablespaces with Automatic Segment Space Management.

Use the following syntax in the <database> block to specify the use of caching for LOBs

```
<ora-db-ddl>
  <ora-lobs caching="true" .../>
</ora-db-ddl>
```

PolicyCenter does not automatically convert LOBs if you change the configuration. You can convert the tables in Oracle and then update the PolicyCenter configuration. PolicyCenter then uses the updated configuration for new objects.

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

See also

- *System Administration Guide*

About 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">
  ...
  <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 if creating a table or index. PolicyCenter does not modify existing tables or indexes. If an upgrade process drops and rebuilds a table, PolicyCenter partitions the table if the table is configured to be partitioned. The schema verifier detects and flags the table if a table is configured as partitioned but is not, or if it is not configured as partitioned but is partitioned.

See also

- *System Administration Guide*

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

```
<database name="PolicyCenterDatabase" dbtype="oracle">
  ...
  <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 use of the `partitioning-type` attribute is mandatory. This attribute takes one of the following values.

LOCAL	The only allowed attribute is the <code>partitioning-type</code> attribute. PolicyCenter partitions the index as the table is partitioned.
HASH	PolicyCenter hash-partitions the index globally on the leading key of the index by using the number of partitions specified or the default number 128.
RANGE	PolicyCenter range-partitions the index by using the <code>range-partitioning-column-list</code> columns and the values specified in the <code>ora-index-range-partition</code> elements under this element. The <code>range-partitioning-column-list</code> element takes a comma-delimited list of columns to use for range-partitioning this index. This element requires the definition of one or more <code>ora-index-range-partition</code> elements. The <code>ora-index-range-partition</code> defines the value range for each partition in <code>value-list</code>. It defines a comma-delimited, ordered list of literal values corresponding to the column list defined in <code>range-partitioning-column-list</code>. Place any single String value inside single quotation marks. Surround the entire list of values by double quotation marks. PolicyCenter uses these defined values in the SQL clause <code>VALUES LESS THAN(value_list)</code>. Do not specify the last range, which is always <code>VALUES LESS THAN (MAXVALUE[, MAXVALUE, ...])</code>. Note: PolicyCenter does not support the use of a date column for range-partitioning of indexes.

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">
...
<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_TPELIST"/>
    <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>
```

See also

- *System Administration Guide*

About Oracle date interval partitioning

It is possible to 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 following attributes:

- Use the `datecolumn` attribute to specify a non-nullable timestamp column for PolicyCenter to 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.

See also

- *System Administration Guide*

About Oracle on Amazon RDS

The RDS option group must have the following:

- LOCATOR
- JVM

The PolicyCenter database user (pcUser) must be granted the `oracle.aurora.security.JServerPermission(LoadClassInPackage.*)` privilege if using the `GWNormalize` Java function to configure search rules.

Guidelines for configuring SQL Server for PolicyCenter

Guidewire recommends that you implement the guidelines in this topic as you install and configure the SQL Server database server to work with PolicyCenter.

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

Note: See the *Supported Software Components* knowledge article for information about which specific SQL Server versions Guidewire supports for PolicyCenter 10.1.2. Visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

Prerequisites to installing PolicyCenter on SQL Server

Create a separate SQL Server 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.

PolicyCenter does not support non-ASCII characters in table names, column names, or other schema object names and qualifiers. Restrict SQL Server database schema object names and qualifiers to ASCII characters.

Decide whether to store other 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, set the `unicodecolumns` attribute to `true` on the `<sqlserver-settings>` subelement of the `<database>` element in `database-config.xml`. Then, during the creation of database tables while starting the application server for the first time, PolicyCenter creates all character columns 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 (those 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 the following location for a full discussion:

<http://msdn.microsoft.com/en-us/library/ms143726.aspx>

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 the following location:

<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.

See also

- *System Administration Guide*

Install PolicyCenter on SQL Server

Procedure

1. Configure SQL Server.
See “Configure SQL Server in Management Studio” on page 42.
2. Create a database for PolicyCenter.
See “Create a PolicyCenter database in SQL Server” on page 42.
3. Modify the `database-config.xml` file so that the PolicyCenter application correctly points to the database.
4. Restart the application server and test by opening PolicyCenter in a browser window.
See “Configuring a database connection” on page 64.
See “Deploying to the application server” on page 83.

Configure SQL Server in Management Studio

Procedure

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 use SQL Server to manage memory.
Consult the following Microsoft documentation or an in-depth discussion of memory options.
[http://msdn2.microsoft.com/en-us/library/ms178067\(d=ide\).aspx](http://msdn2.microsoft.com/en-us/library/ms178067(d=ide).aspx)
Note: Setting the maximum server memory to a specific 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.

Create a PolicyCenter database in SQL Server

About this task

Guidewire requires that you follow these guidelines to create and configure an instance of 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 of the database.

Note: Guidewire recommends that you not share the PolicyCenter database with other applications.

Note: It is possible for a database administrator to write a CREATE DATABASE SQL statement to create the database also.

Procedure

1. If not already open, open SQL Server Management Studio.
2. If creating a new database, do the following. Otherwise, skip to step 3:
 - a. Right-click the **Databases** node and select **New Database**.
 - b. Enter a database name in the **New Database** dialog.
 - c. Click **OK**.
 - d. Skip to step 4.
3. If modifying the model database, do the following:
 - a. Expand **Databases**→**System Databases**,
 - b. Right-click **model** and select **Properties**,
 - c. Continue to step 4.
4. Optionally, create one or more filegroups to support the PolicyCenter logical tablespaces from the **Filegroups** page.
See “Create filegroups in SQL Server” on page 44 for details.
5. Select the **Options** page.
 - a. Choose your database collation if not using the SQL Server server default. The only requirement is that it is a CI (case-insensitive) collation.
 - b. Verify that the values for **Auto Create Statistics** and **Auto Update Statistics** are both set to **True**.
During start up, PolicyCenter checks that these properties are set to **True** and validates that the SQL Server database is case-insensitive.
 - c. Verify that the value of **Auto Shrink** is set to **False**.
Note: If the value of this options is set to **True**, poor performance can result.
 - d. Click **OK**.
6. Right-click **Security** and select **New**→**Login**.
 - a. On the **Login - New** dialog, select **SQL Server Authentication** if not already selected.
 - b. Specify a password and password policy options.
 - c. Click **OK**.
7. In **Object Explorer**, expand the database and open **Security**→**Users**.
 - a. Right-click **Users** and select **New User**.
 - b. Enter pcUser for the **User name**.
 - c. Enter the **Login name** that you created earlier.
 - d. Select db_owner in both **Schemes owned by this user** and **Database role membership** panels.
This action grants ownership of the PolicyCenter database to pcUser.
 - e. Click **OK**.
8. Grant the pcUser the following permissions on each PolicyCenter data management view:
 - view server state
 - view database state

See “Grant Guidewire Data Management view permissions to pcUser” on page 45 for details.

9. Disable the SQL Server autogrowth feature in a production system.

See “Disable the SQL Server autogrowth feature” on page 45 for details.

10. Set the READ_COMMITTED_SNAPSHOT option to on.

PolicyCenter checks for this condition during application start up.

See “Set the READ_COMMITTED_SNAPSHOT option” on page 46 for details.

Note: There is no need to save the READ_COMMITTED_SNAPSHOT query that sets this value.

11. Close SQL Server Management Studio.

About filegroups in SQL Server

You can optionally create one or more filegroups to support the PolicyCenter logical tablespaces. If you choose to use filegroups, create a separate filegroup for each logical tablespace. As you configure the PolicyCenter database connection, you can map the filegroups you create to the PolicyCenter internal logical tablespaces.

For the filegroups, it is possible to use either the same names as the logical tablespace names (with one exception) or to use entirely different names. However, do not use INDEX as a filegroup name. INDEX is a reserved name on SQL Server, so it is not possible to map the logical tablespace INDEX to a physical filegroup of the same name.

See also

- “Configuration options for SQL Server filegroups” on page 66
- “Create filegroups in SQL Server” on page 44

Create filegroups in SQL Server

Before you begin

Review the information in “About filegroups in SQL Server” on page 44 before proceeding with this task.

Procedure

1. If not already open, open SQL Server Management Studio.
2. Select the **Filegroups** page.
3. Create the following file groups:

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 PolicyCenter stores the LOB data in the filegroup mapped to the OP logical tablespace.

Logical Name	Usage
	PolicyCenter uses the LOB filegroup for new tables only. For an existing configuration in which the PolicyCenter schema already exists: • 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.

Grant Guidewire Data Management view permissions to pcUser

About this task

PolicyCenter supports several different data management pages for performance analysis of the application. To use these pages, you must grant the pcUser the following permissions on each PolicyCenter data management view:

- view server state
- view database state

The data management views all start with a `sys.dm_` prefix.

Procedure

1. If not already open, open SQL Server Management Studio.
2. In **Object Explorer**, right-click the database and select **Properties**.
3. Select the **Permissions** page.
4. Select pcUser.
5. Select the check box to grant view database state permission.
6. Click **OK**.
7. Right-click the server and select **Properties**.
8. Select the **Permissions** page.
9. Select the login associated with pcUser.
10. Select the check box to grant view server state permission.
11. Select the check box to grant create any database permission.
This permission allows the `gwb dropDb` command to recreate the database.
12. Click **OK**.

Disable the SQL Server autogrowth feature

About this task

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. You need to disable this feature manually.

Procedure

1. If not already open, open SQL Server Management Studio.
2. Right-click the database and select **Properties**.
3. Select the **Files** page.
 - a. For each database file, click ... in the **Autogrowth** column.
 - b. Click the check box for **Enable Autogrowth** to deselect it.

- c. Click **OK**.
4. Repeat the previous step for each database file.
5. Click **OK** on the **Database Properties** screen.

Set the READ_COMMITTED_SNAPSHOT option

About this task

PolicyCenter requires that the READ_COMMITTED_SNAPSHOT option be set to ON. PolicyCenter checks for this condition during application start up.

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. You can improve performance if you dedicate separate I/O resources to tempdb.

Procedure

1. In SQL Server Management Studio, click **New Query**.
2. In the query pane, enter:

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

3. Click **Execute**.

SQL Server Management Studio shows you that the command completed successfully.

Configuring max degree of parallelism (MAXDOP) in SQL Server

Guidewire recommends setting max degree of parallelism=1 (serial) for regular operations, such as batch processing or online transaction processing (OLTP).

For upgrade and staging table loads, use auto parallelism by setting max degree of parallelism=0. Consider using setting max degree of parallelism=0 when running processes such as database consistency checks, or data distribution reports on a test server.

About index partitioning for SQL Server

The partition-scheme attribute of the <mssql-index-ddl> element specifies the name of the partition scheme to use for the index. For example:

```
<database name="PolicyCenterDatabase" dbtype="sqlserver">
...
<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.

Whenever you partition a SQL Server index, 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.

See also

- *System Administration Guide*

Final SQL Server tests

Before continuing with other installation tasks, perform the following tests on your SQL Server installation:

- 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 requirements

Each developer workstation is a contained environment that includes your company's Guidewire applications and the components needed to configure it. Guidewire recommends that the 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.

Note: See the *Supported Software Components* knowledge article for development workstation system requirements for PolicyCenter 10.1.2. Visit the Guidewire Community and search for knowledge article 1005, "Supported Software Components".

Web client information

PolicyCenter is a web application accessed through a web browser.

Note: See the *Supported Software Components* knowledge article for client system requirements for PolicyCenter 10.1.2. Visit the Guidewire Community and search for knowledge article 1005, "Supported Software Components".

Levels of Guidewire support for web browsers

Guidewire has graded levels of support for web browsers. The following list describes the various grades of support.

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. Guidewire does not support any browser that does not fully support HTML5 and CSS3. 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.

Enable DOM storage in Internet Explorer

About this task

To preserve PolicyCenter user preferences between browser sessions, Guidewire requires that DOM storage be enabled in Internet Explorer. Microsoft enables DOM storage by default. However, if DOM storage is disabled, PolicyCenter displays the following message after you log into PolicyCenter:

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

Procedure

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** check box.
5. Click **OK**.

Installing Java

The PolicyCenter application server and Guidewire Studio require the use of a JVM (Java Virtual Machine).

Supported Java version

The version of the JVM depends on the servlet container and operating system on which the application server runs. See the *Supported Software Components* knowledge article for information on specific version requirements. Visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

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

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

By default, the JVM 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 JVM with Tomcat, add the `-server` flag to `CATALINA_OPTS`.
- If using JVM with WebLogic, 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.

The Dynamic Code Evolution Virtual Machine

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

Guidewire recommends the use of the DCEVM for debugging while developing in the QuickStart environment.

Note: Do not set JAVA_HOME to point to the DCEVM or use the DCEVM JDK to build your project in Guidewire Studio. The DCEVM is intended for use while debugging to hot-swap changes. Using it to build your project (beyond simple changes made while debugging) can cause severe performance problems when doing either incremental or full builds.

Guidewire does not support using the DCEVM for other application servers or in a production environment.

See the *Supported Software Components* knowledge article for information about which specific DCEVM version Guidewire supports for PolicyCenter 10.1.2. Visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

See also

- <http://dcevm.github.io/>

Install the DCEVM

About this task

See the Supported Software Components knowledge article for information about which specific DCEVM version Guidewire supports for PolicyCenter 10.1.2 . Visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

For Java 11

Procedure

1. Visit <http://dcevm.github.io/>.
2. Click the **Java 11** link.
3. Scroll to find the binary for your development platform, and click to download the relevant file:

Windows	java11-openjdk-dcevm-windows.zip
Mac	java11-openjdk-dcevm-osx.tar.gz
Linux	java11-openjdk-dcevm-linux.tar.gz

4. Extract the contents of the downloaded file.
5. Add the DCEVM as an alternate JDK in Studio.
 - a. In Studio, click **File**→**Project Structure**
 - b. Under **Platform Settings**, click **SDKs**.
 - c. Click **Add New SDK** +.
 - d. Select the folder for the DCEVM, and then click **OK**.

For windows: C:\Program Files\Java\dcevm-11.0.1+8

- e. In the **Name** text box, change the name to DCEVM.
- f. Click **OK**.
- g. Click **Run**→**Edit Configurations**.
- h. Under **Application**, click **Servers**.
- i. Next to **JRE**, click **DCEVM**.

Do not set the DCEVM as your project SDK.

For Java 8

Procedure

1. Visit <http://dcevm.github.io/>.
2. Click the link under **Binaries** to download the appropriate DCEVM installer for your java version.
Java 8 update 181
3. Install the DCEVM as an alternative JVM.
 - a. At a command prompt, run `java -jar installer.jar`, where `installer.jar` is the name of the file you downloaded in the previous step.
`java -jar DCEVM-8u181-installer.jar`
 You may need administrator or super user privileges.
 The **Dynamic Code Evolution VM Installer** dialog appears.
 - b. Select the installation directory for the JDK corresponding to the DCEVM version you are installing.
`c:\Program Files\Java\jdk1.8.0_181`
 - c. Click the **Install DCEVM as altjvm** button.
 The **Dynamic Code Evolution VM Installer** dialog displays **Yes** in the **Installed altjvm** column for the selected JDK.
 - d. Click "X" to close the **Dynamic Code Evolution VM Installer** dialog.
4. Edit your configuration in Studio to add `-XXaltjvm=dcevm` to VM options.
 - a. In Studio, at the right hand side of the toolbar, select **Server**.
 - b. Click **Run**→**Edit Configurations**.
 - c. Under **Application**, click **Servers**.
 - d. In **VM options**, add `-XXaltjvm=dcevm` to the end of the line.
 - e. Click **OK**.

Setting environment variables

After you install Java, set environment variables so that PolicyCenter can locate them. You might also need to set an environment variable for your application server.

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	Java 8 <code>C:/Program Files/Java/jdk1.8.0_152</code> Java 11 <code>C:/Program Files/Amazon Corretto/jdk11.0.3_7</code> The Java installer sets JAVA_HOME automatically, but Guidewire recommends that you verify that the installer set the value correctly.
Application server (Tomcat)	CATALINA_OPTS	Specifies the minimum and maximum memory used by Tomcat. For example, the following value for CATALINA_OPTS sets the direct JVM memory allocations to 1024 MB (initially) and 1024 MB (maximum), and also allocates 128 MB of background processing memory: <code>-server -Xms1024M -Xmx1024M</code>

System	Variable	Example Values and Notes
		Make your maximum JVM memory allocation (the <code>-Xmx</code> setting) the maximum likely available memory on the server. Guidewire tests have shown that performance of garbage collections are best if the <code>-Xms</code> and <code>-Xmx</code> are set to the same value. See “Operating system limits on heap size” on page 19 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: <pre>java -X</pre>

Check environment variables

About this task

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

Procedure

1. Open a command prompt.
2. Run the following command:

```
set
```

The command lists the operating system environment variables

Documenting your environment

After establishing 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	
JAVA_HOME	
CATALINA_OPTS	

Installing a PolicyCenter development environment

The development environment enables you to customize PolicyCenter and rapidly view and test your customizations. Do not use a development environment for production.

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 63.

Overview of development environment options

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.
- 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.

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

See also

- “Installing the QuickStart development environment” on page 54
- “Installing PolicyCenter on JBoss in a production environment” on page 83
- “Installing PolicyCenter on WebSphere in a production environment” on page 88
- “Installing PolicyCenter on WebLogic in a production environment” on page 86

Using multiple development instances

Occasionally, you 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.

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 repackage the application.

QuickStart development environment prerequisites

Your environment must meet the minimum requirements for a development workstation. In particular, PolicyCenter requires Java, and Guidewire recommends the use of the DCEVM for debugging in the QuickStart environment. If you install the DCEVM, you will be able to see dynamic changes to Gosu code.

See also

- See the *Supported Software Components* knowledge article for current system and patch level requirements. Visit the Guidewire Community and search for knowledge article 1005, “Supported Software Components”.

About the 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

- *System Administration Guide*

Configuring QuickStart ports

You can specify the port on which the server listens.

The PolicyCenter QuickStart application server listens on port 8180 by default.

The ContactManager QuickStart application server listens on port 8280 by default.

To start the server using a non-default port, use the command `gwb runServer -Dgw.port=nnnn`.

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

About the QuickStart default database

PolicyCenter QuickStart uses an *H2 Database Engine* by default. However, it is also possible to use an Oracle or SQL Server database in your development environment.

Note: To integrate PolicyCenter with ContactManager, see the *Guidewire Contact Management Guide*.

See also

- “Using the QuickStart database” on page 57
- “Using SQL Server or Oracle in a development environment” on page 58
- “Archiving in a development environment” on page 58
- “Install sample data” on page 61
- *System Administration Guide*

Install the bundled PolicyCenter QuickStart application server

Procedure

1. Create an installation directory for PolicyCenter on the host system.
Do not use spaces in the installation directory path. Studio does not run from a directory with a space in its name.
2. Extract the contents of the PolicyCenter Zip file into the installation directory.
Using a third-party tool such as 7-Zip may perform better than using the built-in Windows decompression utility.
3. Open a command prompt to the PolicyCenter installation directory.
4. (Optional) Run the following command:

```
gwb dropDb
```


You need to run this command only if you are reinstalling PolicyCenter and using a QuickStart database.
5. From the same command prompt, run the following command:

```
gwb compile
```


This command copies module resources and JavaScript files to the bundled QuickStart server.
6. (Optional) Configure the location in which PolicyCenter stores the QuickStart database files.
7. Navigate to the root of the PolicyCenter installation directory and open a command prompt.
8. Run the following command:

```
gwb runServer
```


This command starts the PolicyCenter application. After the server starts, you see the following statement in the command window:

```
*****PolicyCenter ready*****
```

9. Open a browser and navigate to the following URL:
`http://localhost:8180/pc`
10. Log into PolicyCenter as the default unrestricted user, using the following default credentials:
Username - su
Password - gw

See also

- See “Modify the QuickStart database file location” on page 58 for details of how to change the QuickStart database file location.
- *Configuration Guide*

QuickStart commands

You launch many PolicyCenter commands by passing arguments to the `gwb` command, located in the PolicyCenter installation directory.

See also

- For a list of `gwb` commands, see “Command reference” on page 123.
- For a list of JVM options that work with the `gwb` commands, see the *System Administration Guide*.

Troubleshooting the QuickStart application server

If you have problems with QuickStart, consider the following issues.

Unable to upgrade

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 the PolicyCenter installation directory and entering the following command:

```
gwb stopServer
```

Then, drop the database with the command `gwb dropDb`.

Port conflicts

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 55

Using 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 uses the QuickStart database.

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.

How to drop the QuickStart database

Open a command prompt in the PolicyCenter installation directory and enter `gwb dropDb`.

Additional quickstart references

For more information about the QuickStart database and tools that you can download to view your schema, see the following web site:

<http://www.h2database.com>

For development environment information, see the *Configuration Guide*.

H2 database configuration

In the base configuration, Guidewire provides several default database configurations for the H2 database in file `database-config.xml`:

- `h2`
- `h2mem`

In Studio, PolicyCenter uses the in-memory version of the H2 database (`h2mem`) while running the server in test mode. Thus, you can start the application server from Studio using either of the following commands from the Studio **Run** menu:

- **Server**
- **TestServer**

If you choose the **Server** command, PolicyCenter starts the server with no named environment, which maps to the `h2` database configuration in file `database-config.xml`:

```
<database name="PolicyCenterDatabase" dbtype="h2">
  <dbcp-connection-pool jdbc-url="jdbc:h2:file:/tmp/guidewire/pc"/>
  <upgrade defer-create-nonessential-indexes="false"/>
</database>
```

If you choose the **TestServer** command, PolicyCenter starts the server with the `h2mem` environment, which maps to the `h2mem` database configuration in file `database-config.xml`:

```
<database name="PolicyCenterDatabase" dbtype="h2" env="h2mem">
  <dbcp-connection-pool jdbc-url="jdbc:h2:mem:/tmp/guidewire/pc"/>
  <upgrade defer-create-nonessential-indexes="false"/>
</database>
```

The `h2mem` configuration starts the application server, which then stores the H2 database in memory. The server then discards the H2 database in memory after you shutdown the server. Thus, the server rebuilds the H2 database every time you start the H2 server in `h2mem` mode.

Modify the QuickStart database file location

About this task

Guidewire uses `/tmp/guidewire` as the database file location and `pc` as the file prefix in the default configuration. You can modify the default database file location and prefix.

Procedure

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

```
gwb studio
```

3. In the **Project** window, expand **configuration**→**config** and open `database-config.xml`.
4. Look for the database block that lists `h2` as the `dbtype`.
5. In this database block, search for the `jdbc-url` attribute value.

The file path after the colon sets the file location. The value after the location sets the file prefix.

In the following example, `tmp/guidewire` is the file location and `pc` is the file prefix.

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

6. Make changes to the file location and prefix as needed for your business needs.
7. In Studio, click **File**→**Save All**.

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. It is possible to use this configuration as a development environment only. Guidewire does not support using the Quickstart server for a production environment.

See also

- For instructions on creating a SQL Server or Oracle database instance, consult “Configuring the database server for Guidewire applications” on page 27.
- For instructions on configuring PolicyCenter to connect to the database, consult “Configuring a database connection” on page 64.

Archiving in a development environment

About this task

If you plan to use archiving in your production environment, Guidewire recommends that you enable archiving in your development environment in order to design and test your custom implementation.

See also

- *Application Guide*
- *Integration Guide*

Archive-related configuration files

The following PolicyCenter configuration elements implement archiving in Guidewire PolicyCenter.

Configuration element	Contains
config.xml	Boolean parameter ArchiveEnabled to enable or disable PolicyCenter archiving
scheduler-config.xml	Schedule on which to run archive batch processing
work-queue.xml	Number of writers to use in executing archive batch processing
Archive batch processing	Batch processing types that manage archive eligibility and archive retrieval
Archive plugins	Integration points necessary for Policy Term archiving

See also

- *Configuration Guide*
- *System Administration Guide*
- *Integration Guide*

Enable archiving in Guidewire PolicyCenter

Procedure

1. In PolicyCenter Studio, navigate to the following location and open file config.xml for editing:

configuration→config

2. Locate configuration parameter ArchiveEnabled and set its value to true.

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

WARNING After you set ArchiveEnabled to true and start the server, you cannot change the value of this parameter again. If you reset the value to false, the server does not start.

3. Navigate to the following location and open file scheduler-config.xml for editing:

configuration→config→scheduler

4. Locate and uncomment the Archive-related code blocks

```
<ProcessSchedule process="ArchivePolicyTerm">
  <CronSchedule dayofmonth="1" hours="1"/>
</ProcessSchedule>

<ProcessSchedule process="RestorePolicyTerm">
  <CronSchedule hours="1" minutes="30"/>
</ProcessSchedule>
```

Guidewire comments out these code block in the base configuration version of this file. To make the archiving schedules active, you must uncomment the code blocks by removing the <!-- and --> tags that surround the code block.

5. Navigate to the following location and open file work-queue.xml for editing:

configuration→config→workqueue

6. Ensure that the number of writer instances is adequate for your PolicyCenter installation.

```
<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>
```

7. Review the default implementations of the following archive plugin interfaces:

configuration→config→Plugins→registry→IArchiveSource

configuration→config→Plugins→registry→IPCArchivingPlugin

Modify and update the plugin implementations to meet your business needs.

8. Save your work.
9. Rebuild and redeploy Guidewire PolicyCenter for your configuration changes to take effect.

See also

- *Configuration Guide*
- *System Administration Guide*
- *Integration Guide*

Disabling Guidewire PolicyCenter archiving

The process to disable Policy Term archiving differs depending on whether you have ever enabled archiving in Guidewire PolicyCenter:

- Archiving never enabled
- Archiving enabled

Archiving never enabled

If you have never enabled archiving in Guidewire PolicyCenter, then you need to do nothing to disable archiving. You simply accept the default archiving configuration, which Guidewire disables in the base configuration by default.

Archiving enabled

After you enable PolicyCenter archiving, you cannot disable archiving entirely unless you drop the application database. This means that once enabled, even after you disable archiving, you still see references to Policy Term archiving within the PolicyCenter user interface.

Do not attempt to disable archiving in a production environment after you have enabled this functionality.

WARNING Do not attempt to reset the value of `ArchiveEnabled` from `true` to `false`. After you set `ArchiveEnabled` to `true` and start the server, you cannot change the value of this parameter again. If you reset the value to `false`, the server does not start. Also, do not remove the archive work queue. Removing this work queue after enabling archiving prevents that application server from starting.

You can actively disable the default, but, unused, configuration elements that PolicyCenter uses in processing archive items. If you choose to disable the unused archiving configuration elements, do the following:

Configuration element	To disable
Batch processing schedule	Comment out the relevant batch processing type in <code>scheduler-config.xml</code> .
Archiving work queue	Set the number of worker instances to 0 in <code>work-queue.xml</code> . Do not attempt to remove or hide this work queue after you enable PolicyCenter archiving.
Archive plugin implementations	Select the Disabled box in the archive plugin editor for the relevant archive plugin implementation. You access the archive plugin editor in Studio in the following location: configuration→config→Plugins→registry ...

Rebuild and redeploy Guidewire PolicyCenter for your configuration changes to take effect.

See also

- *Configuration Guide*
- *System Administration Guide*

- *Integration Guide*

Install sample data

About this task

PolicyCenter includes sample data for use in training, configuration and testing. The PolicyCenter server must be in development mode to be able to load sample data. The QuickStart server is always in development mode.

IMPORTANT Guidewire expressly does not support any attempt to load the default sample data into a production system.

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

If you have previously loaded a sample data set, you must drop the PolicyCenter database before it is possible to load a different sample data set.

Procedure

1. Log into PolicyCenter using an administrative user 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 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.
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 displays a message after the sample data import completes.

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

See also

- For instructions on how to import or export administrative data, see the *System Administration Guide*.
- “Using Gosu to configure sample data” on page 62
- “Verify sample data changes in PolicyCenter” on page 62

Using Gosu to configure sample data

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 account data in the small sample data set, edit `SmallSampleAccountData.gs`.

Guidewire recommends the following guidelines for editing the data:

- The main `SampleData` class merely invokes the appropriate data collections. Put the data in there.
- While 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.

Verify sample data changes in PolicyCenter

Procedure

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, for example:

```
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. Click **Internal Tools**.
8. Click **PC Sample Data**.
9. Click **Load** for the **Tiny** dataset.
10. Log out and log back in as the new user to verify the user has been created, indicating your new sample data has loaded correctly.

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.

See also

- “Installing a PolicyCenter development environment” on page 53

Installing Guidewire PolicyCenter

Guidewire packages the PolicyCenter application as a Zip file. This file contains tools and files necessary to build a WAR or EAR file to install on an application server. It also 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.

Guidewire recommends that you maintain your PolicyCenter configuration files in a change control system such as Perforce or SVN.

See also

- For an overview of the directories included with PolicyCenter, see the *Configuration Guide*.

Unpack the PolicyCenter configuration files

Procedure

1. If you have not already done so, create an installation directory for PolicyCenter.
Do not use spaces in the installation directory path. Studio does not run from a directory with a space in its name.
2. Extract the contents of the PolicyCenter Zip file into the installation directory.
Using a third-party tool such as 7-Zip may perform better than using the built-in Windows decompression utility.

3. Do one of the following:

- If using a change control system such as Perforce or SVN to manage PolicyCenter configuration files, add the files in the PolicyCenter modules/configuration directory to change control at this point.
- If not using a version control system, make a read-only copy of the PolicyCenter directory. The use of back-up copy enables you to recover quickly from accidental changes that can prevent PolicyCenter from starting.

Result

At this point, you have a full set of PolicyCenter configuration files.

Configuring a database connection

The topics in this section discuss how to configure a database connection.

After you complete the configuration of the database connection, proceed to “Deploying to the application server” on page 83.

About the database element

File database-config.xml stores connection and configuration parameters for the PolicyCenter database. Set database connections by uncommenting and modifying the appropriate sample <database> element in database-config.xml, accessible within Guidewire Studio under **configuration**→**config**.

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

```
<database name="string" env="string" dbtype="oracle|sqlserver"
  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.

File database-config.xml contains a single root-level <database> element that takes the following attributes.

Attribute	Re- quired	Default	Sets
addforeignkey	No	true	Used only for development and testing. Do not use this attribute in production.
checker	No	false	IMPORTANT Guidewire deprecates this attribute and recommends that you not use it. Boolean value that specifies whether PolicyCenter runs consistency checks before it starts. - Development environments – For development environments with small data sets, you can enable consistency checks to run each time the PolicyCenter server starts. Set the value of checker in the database block to true to enable checks on server startup. - Production environments – Running consistency checks upon server startup can take a long time, impact performance severely, and possibly time out on very large datasets. Set the value of checker in the database block to false to disable checks on server startup. Recommendations: - True – Guidewire recommends that you only set checker to true in development environments with small test data sets. - False – Guidewire recommends that you set checker to false under most circumstances. See also <i>System Administration Guide</i>
dbtype	Yes	...	Database type, either h2 (for the QuickStart database), oracle or sqlserver.
env	No	...	Optional environment variable. Use of the env attribute to set a server environment enables you to provide different database configurations for different server environments. For example, you can set up different database configurations for a production environment and a test environment. See the <i>System Administration Guide</i> for more information.
name	Yes	...	A string identifying the database for which PolicyCenter uses this connection specification.
printcommands	No	true	Boolean value that specifies whether the server prints database upgrade messages to the console upon startup. By default, Guidewire sets the value of printcommands to true in the base configuration. Do not set printcommands to false in a production environment.
versioncheckonly	No	false	Boolean value that specifies whether the PolicyCenter server runs only database version checks at startup, without performing any actual database upgrade steps. See also <i>Checking the database before upgrade</i> in the <i>Upgrade Guide</i>

See also

- “Configuring the database server for Guidewire applications” on page 27
- System Administration Guide*

Improving screen performance in PolicyCenter

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

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

Note: Guidewire requires that you provide a value for attribute `numbuckets`. The default value for the number of buckets is 254 for the retired and subtype columns. For all other columns, PolicyCenter uses 75, the database default.

Mapping logical tablespaces to physical tablespaces

It is possible to map the logical tablespaces required by PolicyCenter to either Oracle tablespaces or SQL Server filegroups using database configuration file `database-config.xml`:

- For Oracle, Guidewire requires that you map the Guidewire logical tablespaces to the Oracle tablespaces.
- For SQL Server, mapping the Guidewire logical tablespaces to filegroups is optional. Create these physical tablespaces or filegroups while you set up your database.

See also

- See “Configuring the database server for Guidewire applications” on page 27 for information on creating physical tablespaces or filegroups for your database.

Configuration options for Oracle tablespaces

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>
```

Configuration options for SQL Server filegroups

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>

```

Configuration options for individual database tables

You can configure DDL options for individual tables and indexes in the database configuration file, `database-config.xml`. It is not possible to specify the DDL attributes of the primary key backing index (on column named with suffix ID) in the configuration file. These attributes use the DBMS default values.

This topic shows valid syntax in `database-config.xml` for configuring DDL options for each database type. All options are shown for reference.

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-lobs 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-lobs 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 database compression” on page 29
- “Guidelines for configuring Oracle for PolicyCenter” on page 34
- “Guidelines for configuring SQL Server for PolicyCenter” on page 41

Configuration options for database table groups

Table groups specify a set of tables on which to run database consistency checks. Their use is optional. Thus, you can define zero or more table groups.

As 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.

To define a table group, add a `<tablegroup>` element to the database element in `database-config.xml`. The `<tablegroup>` element has the following attributes:

env	Specifies the environment for the table group. You can have different table groups set up for different environments.
name	Identifies the table group.
tables	Defines which tables are in the table group. Use a comma-separated list to specify the tables, for example:

```

<database>
  ...
  <tablegroup name="MyTables" tables="pc_1, pc_2, pc_3"/>
  ...
</database>

```

See also

- *System Administration Guide*

Specifying additional database parameters

To pass additional parameters required for your database connection, specify one or more name-value `<param>` elements in database configuration file `database-config.xml`. Several parameters configure the connection pool, if you are using PolicyCenter to manage the connection pool rather than a JNDI data source managed by the application server.

See also

- See the *System Administration Guide* for a description of the database configuration parameters that control the connection pool.

The JDBC URL format

In database configuration file `database-config.xml`, 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 the application server.

If you want to use a JNDI data source managed by the application server, skip to “Configure PolicyCenter to use a direct JNDI data source” on page 71.

JDBC URL format for Oracle

In database configuration file `database-config.xml`, the JDBC URL for a standalone Oracle instance uses one 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 server name can be specified using the computer name or IP address.

JDBC URL format for SQL Server

In database configuration file `database-config.xml`, the JDBC URL for SQL Server uses the following format:

```
<dbcp-connection-pool jdbc-url="jdbc:sqlserver://serverName[:port];
databaseName=pc;user=pcUser;password=password
[;applicationName=applicationName]"/>
```

The example code shows optional parameters in brackets.

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. If 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 in the thread identification information. 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.

Obfuscating the database password

It is likely that you do not want to expose the password in the JDBC URL in database configuration file `database-config.xml`. For this reason, Guidewire provides the following alternatives for hiding the database password:

Use a password file	See "Use a password file to obfuscate the database password" on page 70
Implement the Database Authentication Plugin	To hide the password in the JDBC URL in <code>database-config.xml</code>, Guidewire provides a default implementation of the <code>DBAuthenticationPlugin</code> authentication plugin. The use of this plugin implementation provides a higher level of security than the use of an external file to store the password. 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 the <code>DBAuthenticationPlugin</code> plugin in your environment, see the <i>Integration Guide</i>.
Use a JNDI data source	You can configure PolicyCenter to use a JNDI data source for your database connection on any of the following application servers: - JBoss - WebLogic - WebSphere The JNDI data source uses a Java 2 Connector (J2C) authentication alias to store the user name and password. See also "Configure PolicyCenter to use a direct JNDI data source" on page 71.

Use a password file to obfuscate the database password

About this task

To hide the password in the JDBC URL, you can place the password in an external file and reference this file from the `database-config.xml` file.

Procedure

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.

Result

After you complete this process, your database specification looks similar to the following:

Oracle

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

SQL Server

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

About SQL Server JDBC logging

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

It is possible to turn on Microsoft JDBC driver logging at PolicyCenter startup 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 the following Microsoft site:

[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)

If you do not specify the location of the trace file, it defaults to the following location:

`C:/temp/msjdbctrace%u.log`

Refer to the following web site for documentation on the trace file specification:

<http://docs.oracle.com/javase/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.

Guidewire provides a means to manage JDBC Driver logging in the PolicyCenter Server Tools **Info Pages**→**Microsoft JDBC Driver Logging** screen. Use this screen to start and stop Microsoft driver logging on a running application server. Using this screen is a better option if tracing a particular operation, in order to minimize system impact and size of the trace file.

See also

- *System Administration Guide*

Using a JNDI data source

It is possible for PolicyCenter to use a Java Naming and Directory Interface (JNDI) data source managed by any of the following application servers:

- JBoss
- Tomcat
- WebLogic
- WebSphere

The use of JNDI data source 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.

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

```
INFO Looking up JNDI datasource 'jdbc/pcDataSource'...
```

Important caveats

1. Always refer to the latest documentation from third party vendors.
2. Guidewire supports JNDI using the drivers bundled with PolicyCenter only. Guidewire does not support the XA versions of a data source.
3. If using a JNDI data source, create the JNDI data source on the application server before deploying PolicyCenter to the application server.

Configure PolicyCenter to use a direct JNDI data source

Procedure

1. Open Guidewire Studio™ for PolicyCenter.
2. In the **Project** window, expand **configuration**→**config**.
3. Open file `database-config.xml`.
4. Remove the `<dbcp-connection-pool>` element.
5. Add a `<jndi-connection-pool>` element and specify the JNDI name you assign to the data source as a `datasource-name` attribute.

After you complete this process, the `<database>` element looks similar to the following:

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

6. Close and save the `database-config.xml` file.
7. Rebuild and install the PolicyCenter application EAR or WAR file.

Configure PolicyCenter to use an indirect JNDI data source

Procedure

1. In the PolicyCenter Studio **Project** window, expand **configuration**→**config**:
2. Open file `database-config.xml`.

3. Remove the `<dbcp-connection-pool>` element.
4. Add a `<jndi-connection-pool>` element and specify the JNDI name you assign to the data source as a `datasource-name` attribute.

After you complete this process, the `<database>` element looks similar to the following:

```
<database name="PolicyCenterDatabase" dbtype="oracle|sqlserver">
  <jndi-connection-pool datasource-name="java:comp/env/jdbc/DataSourceName"/>
  ...
</database>
```

5. Close and save the `database-config.xml` file.
6. Rebuild and install the PolicyCenter application EAR or WAR file.

Create an Oracle JNDI data source on JBoss

Procedure

1. Start the JBoss application server.
2. Log in to the JBoss Admin Console.
3. Click the **Deployments** tab for the domain hosting PolicyCenter.
JBoss does not show a domain for standalone installations.
4. Click the **Add** button.
The **New Deployment** wizard starts.
5. Select **Upload a new deployment** and click **Next**.
6. Click **Choose File**.
7. Browse to the appropriate directory for your Java version, under `PolicyCenter/admin/lib`.

Java 8	<code>PolicyCenter/admin/lib/java8</code>
Java 11	<code>PolicyCenter/admin/lib/java11</code>
8. Select the `ojdbc-<version>.jar` file and click **Next**.
9. Check the **Enable** check box (if not checked) and click **Finish**.
A confirmation message box confirms that you deployed `ojdbc-<version>.jar` successfully.
10. Click the **Configuration** tab.
11. Click **Subsystems**→**Datasources**→**Non-XA**.
12. Click **Add**.
The **Create Datasource** wizard starts.
13. Click **Subsystems**→**Connector**→**Datasources**.
14. Select **Oracle Datasource** and click **Next**.
15. Enter a **Name** for the data source.
16. Enter a **JNDI Name**. The JNDI name must use the pattern `java:jboss/datasources/name`, where *name* is the name of the data source from the previous step.
The JNDI name must match the value of the `datasource-name` attribute of the `<jndi-connection-pool>` element in the `database-config.xml` file.
17. Click **Next**.
18. Select the Oracle JDBC driver, (exactly as in step step8). Keep default values for other fields and click **Next**.
19. Enter the **Connection URL**, **Username**, and **Password** and click **Next**.
20. Confirm information provided on the **Summary** page and click **Finish**.
Message "Added datasource *name*" is shown. In case of error, the new data source was not added, and you must repeat steps starting from step step12.

21. Select the newly added data source and click **Test Connection**.

If you receive a message that the JDBC connection failed, check your connection settings.

Create a SQL Server JNDI data source on JBoss

Procedure

1. Start the JBoss application server.
2. Log in to the JBoss Admin Console.
3. Click **Deployments** for the domain hosting PolicyCenter.
JBoss does not show a domain for standalone installations.
4. Click the **Add** button.
The **New Deployment** wizard starts.
5. Select **Upload a new deployment** and click **Next**.
6. Click the **Choose File** button.
7. Browse to the appropriate directory for your Java version, under PolicyCenter/admin/lib.

Java 8	PolicyCenter/admin/lib/java8
Java 11	PolicyCenter/admin/lib/java11
8. Select the `mssql-jdbc-<version>.jar` file and click **Next**.
9. Check the **Enable** check box (if not checked) and click **Finish**.
A confirmation message box confirms that you deployed `mssql-jdbc-<version>.jar` successfully.
10. Click the **Configuration** tab.
11. Click **Subsystems**→**Datasources**→**Non-XA**.
12. Click **Add**.
The **Create Datasource** wizard starts.
13. Select **Microsoft SQLServer Datasource** and click **Next**.
14. Enter a **Name** for the data source.
15. Enter a **JNDI Name**. The JNDI name must use the pattern `java:jboss/datasources/name`, where *name* is the name of the data source from the previous step.
The JNDI name must match the value of the `data source-name` attribute of the `<jndi-connection-pool>` element in the `database-config.xml` file.
16. Click **Next**.
17. Select the SQL Server JDBC driver, (exactly as in step step8). Keep default values for other fields, and click **Next**.
18. Enter the **Connection URL**, **Username**, and **Password** and click **Next**.
19. Confirm information provided on the **Summary** page and click **Finish**.
Message "Added datasource *name*" is shown. In case of error, the new data source was not added and you must repeat steps starting from step12.
20. Select the newly added data source and click **Test Connection**.
If you receive a message that the JDBC connection failed, check your connection settings.

Create an Oracle JNDI data source on Tomcat

About this task

Guidewire bundles the supported version of the Oracle JDBC Thin Driver in the `ojdbc-<version>.jar` JAR file.

Refer to the following web site for more information about configuring the data source:

<http://commons.apache.org/proper/commons-dbcp/configuration.html>

Procedure

1. Browse to the appropriate directory for your Java version, under PolicyCenter/admin/lib.

Java 8 PolicyCenter/admin/lib/java8

Java 11 PolicyCenter/admin/lib/java11

2. Copy the `ojdbc-<version>.jar` file to the Tomcat lib directory.

This JAR file contains the APIs for connecting to the PolicyCenter database.

3. 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" maxTotal="20" maxIdle="10" maxWaitMillis="-1"/>
```

4. 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>
```

5. Restart the Tomcat instance.

Create a SQL Server JNDI data source on Tomcat

Procedure

1. Browse to the appropriate directory for your Java version, under PolicyCenter/admin/lib.

Java 8 PolicyCenter/admin/lib/java8

Java 11 PolicyCenter/admin/lib/java11

2. Copy the `mssql-jdbc-<version>.jar` JAR file to the Tomcat lib directory.

This JAR file contains the APIs for connecting to the PolicyCenter database.

3. 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" maxTotal="20" maxIdle="10"
  maxWaitMillis="-1"/>
```

4. 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>
```

5. Restart the Tomcat instance.

Create an Oracle JNDI data source on WebLogic

About this task

Guidewire bundles the supported version of the Oracle JDBC Thin Driver in the `ojdbc<version>.jar` JAR file.

IMPORTANT You must use this Guidewire bundled version of the Oracle JDBC Thin Driver and not the default driver provided by WebLogic.

Procedure

1. Browse to the appropriate directory for your Java version, under `PolicyCenter/admin/lib`.

Java 8	<code>PolicyCenter/admin/lib/java8</code>
Java 11	<code>PolicyCenter/admin/lib/java11</code>
2. Copy the `ojdbc<version>.jar` file to the WebLogic `$DOMAIN/lib` directory.
3. Backup and remove the `com.oracle.db.jdbc7-dms.jar` from the `$WLS_INSTALL/oracle_common/modules/features` folder.
4. For Java 11 only: backup and remove the three `ojdbc8*.jar` drivers from the `$WLS_INSTALL/oracle_common/modules/oracle.jdbc` folder. (You can ignore the `ojdbc7` drivers.)
Delete `ojdbc8.jar`, `ojdbc8_g.jar`, and `ojdbc8dms.jar` from the `c:/example/weblogic-12.2.1/oracle_common/modules/oraclejdbc` directory.
5. Ensure that `ojdbc<version>.jar` is not in any part of the `CLASSPATH`.
6. In the `$WLS_INSTALL/oracle_common/common/bin` directory, edit `commExtEnv.sh` and add `ojdbc<version>.jar` to the beginning of `WEBLOGIC_CLASSPATH`.
7. Open the WebLogic Server Administration Console.
8. Choose **Service**→**Data Source**.
9. Click **Lock & Edit**.
10. Click **New** to create a Generic Data Source.
11. 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/`.
12. Select **Oracle** as the **Database Type**.
13. Select **Oracle's Driver (Thin)** for **Service Connections**.
14. Select **Versions:Any** for **Database Driver** and click **Next**.
15. Fill in connection properties for your environment and click **Next**.
16. Uncheck **Supports Global Transactions**, click **Next** and complete the data source creation.
17. Deploy the EAR file for PolicyCenter which has appropriate JNDI setting in `database-config.xml`.
18. Start PolicyCenter.

A log similar to the following example demonstrates that the created JNDI is being used:

```
example-t46 2018-08-28 13:58:03,608 INFO Finished security config
example-t46 2018-08-28 13:58:03,608 INFO Starting database ClaimCenterDatabase on env null
example-t46 2018-08-28 13:58:03,608 INFO Using JNDI data source
example-t46 2018-08-28 13:58:03,608 INFO Looking up JNDI datasource 'jdbc/cliCCDataSource'...
example-t46 2018-08-28 13:58:03,608 INFO Using JNDI datasource jdbc/cliCCDataSource
```

Create a SQL Server JNDI data source on WebLogic

Procedure

1. Browse to the appropriate directory for your Java version, under PolicyCenter/admin/lib.

Java 8 PolicyCenter/admin/lib/java8

2. Copy the `mssql-jdbc-<version>.jar` JAR file to the `server/lib` directory within the WebLogic home. This JAR file contains the APIs for connecting to the PolicyCenter database.
3. Add the `mssql-jdbc-<version>.jar` file to the classpath of the WebLogic domain by doing one of the following:

Modify classpath for single domain	• Open the <code>WL_HOME/common/bin/commEnv</code> script appropriate to your operating system in a text editor. • Prepend the absolute path to the <code>mssql-jdbc-<i><version></i>.jar</code> file, including the file name, to the <code>WEBLAGIC_CLASSPATH</code> environment variable.
------------------------------------	---

Modify classpath for multiple domains	• Open the <code>setDomainEnv</code> script appropriate to your operating system in a text editor. • Prepend the absolute path to the <code>mssql-jdbc-<i><version></i>.jar</code> file to the <code>PRE_CLASSPATH</code> environment variable.
---------------------------------------	--

4. Restart the WebLogic server for the change to take effect.
5. Open the WebLogic Server Administration Console.
6. Click **Service**→**Data Source**.
7. Click **New** to create a Generic Data Source.
8. 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.
9. Select **MS SQL Server** as the **Database Type**.
10. Select **Other** as the **Database Driver**, and click **Next**.
11. Uncheck **Supports Global Transactions**, and click **Next**.
12. Specify connection properties for the SQL Server database, and click **Next**.
13. Enter `com.microsoft.sqlserver.jdbc.SQLServerDriver` for the **Driver Class Name**.
14. Specify the **URL** using the following format:

```
jdbc:sqlserver://servername:port;datasasename=dbname;user=username
```

15. Fill in the connection properties for your environment:
 - **Unicode database** – Set the value of `sendStringParametersAsUnicode` to `true`.
 - **Single-byte database** – Set the value of `sendStringParametersAsUnicode` to `false`.
16. Click **Test Configuration**.
If you configured the connection properly and the database is running, WebLogic displays the message “Connection test succeeded.”
17. Click **Next**.
18. Select the targets that use the data source.
19. Click **Finish**.

The WebLogic Server Administration Console returns you to the **Summary of JDBC Data Sources** screen.

20. Click **Activate Changes**.

WebLogic displays the message “All changes have been activated. No restarts are necessary.”

Create an Oracle JNDI data source on WebSphere

About this task

Creating an Oracle JNDI data source on WebSphere is a multistep process.

Copy the Oracle JDBC driver to WebSphere

Before you begin

Before starting this procedure, review “Create a SQL Server JNDI data source on WebSphere” on page 79.

About this task

Guidewire bundles the supported version of the Oracle JDBC Thin Driver in the `ojdbc-<version>.jar` JAR file.

Procedure

1. Browse to the appropriate directory for your Java version, under `PolicyCenter/admin/lib`.

Java 8 `PolicyCenter/admin/lib/java8`

2. Copy the `ojdbc-<version>.jar` file to the WebSphere `WAS_HOME/lib/ext` directory.
This JAR file contains the APIs for connecting to the PolicyCenter database.

Next steps

After completing this procedure, proceed to “Create the Oracle JDBC provider” on page 77.

Create the Oracle JDBC provider

Before you begin

Before starting this procedure, complete “Copy the Oracle JDBC driver to WebSphere” on page 77.

Procedure

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 `ojdbc-<version>.jar`.
Set the value to the `WAS_HOME/lib/ext` path in which you copied `ojdbc-<version>.jar` earlier.
12. Click **Next**.
13. Review the **Summary** screen and do one of the following:
 - 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** screen. At this point, you have completed the creation of the new provider.

Next steps

After completing this procedure, proceed to “Create the Oracle JNDI data source” on page 78.

Create the Oracle JNDI data source

Before you begin

Before proceeding, complete “Create the Oracle JDBC provider” on page 77.

Procedure

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 apply your changes to the master configuration.
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** screen.
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 file `database-config.xml`. See “Using a JNDI data source” on page 71.
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**.
16. Review the information on the **Summary** screen and do one of the following:
 - Click **Previous** if you need to make changes.
 - Otherwise, click **Finish**.
17. Click **Save** to apply your changes to the master configuration.

Next steps

After completing this procedure, proceed to “Configure the Oracle data source properties” on page 79.

Configure the Oracle data source properties

Before you begin

Before proceeding, complete “Create the Oracle JNDI data source” on page 78.

Procedure

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** screen, 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.

Next steps

After completing this procedure, proceed to “Test the Oracle JNDI connection” on page 79.

Test the Oracle JNDI connection

Before you begin

Before proceeding, complete “Configure the Oracle data source properties” on page 79.

Procedure

1. Open the WebSphere Administrative Console if it is not already open.
2. Select **Resources**→**JDBC**→**Data sources**.
3. Select the check box next to the PolicyCenter data source.
4. Click **Test Connection** to verify that the connection works.

Create a SQL Server JNDI data source on WebSphere

About this task

Creating a SQL Server JNDI data source on WebSphere is a multistep process.

Copy the SQL Server JDBC driver to WebSphere

Before you begin

Before proceeding, review “Create a SQL Server JNDI data source on WebSphere” on page 79.

Procedure

1. Browse to the appropriate directory for your Java version, under PolicyCenter/admin/lib.

Java 8 PolicyCenter/admin/lib/java8

2. Copy the mssql-jdbc-*<version>*.jar JAR file to the WebSphere lib/ext directory.

This JAR file contains the APIs for connecting to the PolicyCenter database.

Next steps

After completing this procedure, proceed to “Create the SQL Server JDBC provider” on page 80.

Create the SQL Server JDBC provider

Before you begin

Before proceeding, complete “Copy the SQL Server JDBC driver to WebSphere” on page 79.

Procedure

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 mssql-jdbc-*<version>*.jar.
Set the value to the WebSphere lib/ext directory, for example:

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

You can ignore the greyed out **Class path** box that lists the JAR name as sqljdbc.jar instead of mssql-jdbc-*<version>*.jar.

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

12. Click **Next**.
13. Review the **Summary** screen and do one of the following:
 - 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** screen. At this point, you have completed the creation of the new provider.
15. Select the JDBC provider you just created.
16. Edit the value of **Class path** and change it from \${MICROSOFT_JDBC_DRIVER_PATH}/sqljdbc.jar to \${MICROSOFT_JDBC_DRIVER_PATH}/mssql-jdbc-*<version>*.jar.
17. Click **OK**.
18. Click **Save** to apply your changes to the master configuration.

Next steps

After completing this procedure, proceed to “Create the SQL Server data source” on page 81

Create the SQL Server data source

Before you begin

Before proceeding, complete “Create the SQL Server JDBC provider” on page 80.

Procedure

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, for example pcAlias.
User ID	A string specifying the user name.
Password	A string specifying the password.

- d. Click **OK**.
 - e. Click **Save** to apply your changes to the master configuration.
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** screen.
 7. Click **New** to create a new data source and enter a value for **JNDI name**.

For example, enter something 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 “Configure PolicyCenter to use a direct JNDI data source” on page 71.
 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 **Component-managed authentication alias**.
 15. Select an authentication alias for **Container-managed authentication alias**.
 16. Click **Next**.
 17. Review the information on the **Summary** screen. Do one of the following:
 - Click **Previous** if you need to make changes.
 - Otherwise, click **Finish**.
 18. Click **Save** to apply your changes to the master configuration.

Next steps

After completing this procedure, proceed to “Configure the SQL Server data source properties” on page 82.

Configure the SQL Server data source properties

Before you begin

Before proceeding, complete “Create the SQL Server data source” on page 81.

Procedure

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 value of **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. Verify the value of property **sendStringParametersAsUnicode**.
To change the value, click the property name:
 - If you are using unicode columns (nvarchar), set this value to true.
 - If you are not using nvarchar, but single byte varchar columns, set this value to false.Your application server will not start up if this setting is incorrect.
12. At the top of the **Custom properties** screen, click **New**.
 - a. Enter **commitOrRollbackOnCleanup** in the **Name** field.
 - b. Enter **rollback** in the **Value** field.
 - c. Click **OK**.
13. Click **Save** to apply your changes to the master configuration.

Next steps

After completing this procedure, proceed to “Test the SQL Server JNDI connection” on page 82.

Test the SQL Server JNDI connection

Before you begin

Before proceeding, complete “Configure the SQL Server data source properties” on page 82.

Procedure

1. Open the WebSphere Administrative Console if it is not already open.
2. Select **Resources**→**JDBC**→**Data sources**.
3. Select the check box next to the PolicyCenter data source.
4. Click **Test Connection** to verify that the connection works.

Deploying to the application server

Deploying PolicyCenter to an application server 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 on how to create a development PolicyCenter environment, see “Installing a PolicyCenter development environment” on page 53.

Installing PolicyCenter on JBoss in a production environment

To use JBoss as the PolicyCenter application server, you must create and deploy a JBoss-specific PolicyCenter WAR package file to the application server. Generating this WAR file is a multistep process.

Note: For instructions on how to deploy the PolicyCenter WAR file, refer to the JBoss Enterprise Application Platform *Administration and Configuration* Guide.

Set Java options for JBoss

Before you begin

Before proceeding, review “Installing PolicyCenter on JBoss in a production environment” on page 83.

About this task

If you are running JBoss in a 64-bit JVM, it is possible that you need to increase the heap size to prevent JBoss from running out of memory. The specific values used for JAVA_OPTS in this example may not be suitable for your environment. Contact Guidewire Support if you need assistance.

Note: You can specify Java options in the JBoss `standalone.conf.bat` (Windows) or `standalone.conf` (Linux) file in the `EAP_HOME/bin` directory, as this procedure describes, or by creating a `JAVA_OPTS` environment variable.

Procedure

1. Open the JBoss file for your operating system.

```
bin/standalone.conf.bat  
bin/standalone.conf
```

2. Add lines similar to the following:

```
set JAVA_OPTS=-Xms512M -Xmx2G -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -  
Dorg.jboss.resolver.warning=true -server
```

```
set JAVA_OPTS=%JAVA_OPTS% -Xbootclasspath/a:%JAVA_HOME%\lib\tools.jar
```

3. Save the file.
4. Using Guidewire Studio, add the following entry to `jboss-deployment-structure.xml` next to the existing `sun/net` paths in that file (near line 21): `<path name="com/sun/javadoc"/>`

Next steps

After completing this procedure, proceed to “Add a servlet definition for JBoss” on page 83.

Add a servlet definition for JBoss

Before you begin

Before proceeding, complete “Set Java options for JBoss” on page 83.

Procedure

1. Launch Guidewire Studio:
 - a. Open a command prompt and navigate to the PolicyCenter installation directory.
 - b. Execute the following command:


```
gwb studio
```
2. In the Studio **Project** window, expand **configuration→deploy→WEB-INF**:
 - a. Open file `web.xml`:
 - b. Add servlet definitions as needed.
 - c. Add a servlet-mapping definition for each servlet that you add.
Review the defined servlets for examples.
3. Save your changes.

Next steps

Review the post-installation tasks in “Additional PolicyCenter setup tasks” on page 91. After completing this procedure, proceed to “Generate the PolicyCenter WAR file for JBoss” on page 84.

Generate the PolicyCenter WAR file for JBoss

Before you begin

Before proceeding, complete “Add a servlet definition for JBoss” on page 83.

About this task

You can build the PolicyCenter WAR file for JBoss with or without JDBC drivers by using one of the following build commands:

<code>warJbossDbcp</code>	Builds the WAR file with JDBC drivers. PolicyCenter manages the database connection pool.
<code>warJbossJndi</code>	Builds the WAR file without JDBC drivers. JBoss manages a JNDI database connection pool.

Procedure

1. Open a command prompt and navigate to the PolicyCenter installation directory.
2. Execute one of the following commands:


```
gwb warJbossDbcp
gwb warJbossJndi
```

Both commands generate the PolicyCenter WAR file in the PolicyCenter `dist/war/JbossJndi` or `dist/war/JbossDbcp` directory.
3. Deploy the package to JBoss according to the instructions for deploying an application included with JBoss.

Next steps

After completing this procedure, proceed to “Start PolicyCenter on JBoss” on page 109.

Installing PolicyCenter on Tomcat in a production environment

To use Apache Tomcat as the application server, you must create and deploy a Tomcat-specific WAR package file to that application. Generating the WAR file is a multistep process.

Add servlet definitions for Tomcat

Before you begin

Before proceeding, review “Installing PolicyCenter on Tomcat in a production environment” on page 84.

About this task

It is possible to add definitions for additional servlets to the PolicyCenter `web.xml` file.

Procedure

1. In the Studio **Project** window, expand **configuration**→**deploy**→**WEB-INF**:
 - a. Open file `web.xml`:
 - b. Add a `servlet-mapping` definition for each servlet that you want to add.
Review the defined servlets for an example.
2. Save `web.xml`.

Next steps

After completing this procedure, proceed to “Generate and deploy the PolicyCenter WAR file for Tomcat” on page 85.

Generate and deploy the PolicyCenter WAR file for Tomcat

Before you begin

Before proceeding, complete “Add servlet definitions for Tomcat” on page 85.

About this task

You can build the PolicyCenter WAR file for Tomcat with or without JDBC drivers by using one of the following commands:

<code>warTomcatDbcP</code>	Builds the WAR file with JDBC drivers. PolicyCenter manages the database connection pool.
----------------------------	---

<code>warTomcatJndi</code>	Builds the WAR file without JDBC drivers. JBoss manages a JNDI database connection pool.
----------------------------	--

Procedure

1. Open a command prompt and navigate to the PolicyCenter installation directory.
2. Execute one of the following commands:

```
gwb warTomcatDbcP
gwb warTomcatJndi
```

Both commands generate the PolicyCenter WAR file in the PolicyCenter `dist/war` directory.
3. Before you deploy PolicyCenter to Tomcat, verify that environment variable `CATALINA_OPTS` has the following value:

```
-Xms1024M -Xmx2G
```

4. Deploy the package to Tomcat by copying the `pc.war` file to the `webapps` directory in your Tomcat server.

Result

As 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.

Next steps

After completing this procedure, review the post-installation tasks in “Additional PolicyCenter setup tasks” on page 91. Then, proceed to “Start PolicyCenter on Tomcat on Windows” on page 109.

Installing PolicyCenter on WebLogic in a production environment

To use WebLogic as the PolicyCenter application server, you must create and deploy a WebLogic-specific EAR package file. Generating the EAR file is a multistep process.

Add a servlet definition for WebLogic

Before you begin

Before proceeding, review “Installing PolicyCenter on WebLogic in a production environment” on page 86.

About this task

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

Procedure

1. In the Studio **Project** window, expand **configuration→deploy→WEB-INF**:
 - a. Open file `web.xml`:
 - b. Add a servlet-mapping definition for each servlet that you want to add.
Review the defined servlets for an example.
2. Save `web.xml`.

Next steps

After completing this procedure, proceed to “Enable HTTP authentication on WebLogic” on page 86.

Enable HTTP authentication on WebLogic

Before you begin

Before proceeding, complete “Add a servlet definition for WebLogic” on page 86.

About this task

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>
```

Next steps

After completing this procedure, proceed to “Generate the PolicyCenter EAR file for WebLogic” on page 86.

Generate the PolicyCenter EAR file for WebLogic

Before you begin

Before proceeding, complete “Enable HTTP authentication on WebLogic” on page 86.

About this task

You can build the PolicyCenter EAR file for WebLogic with or without JDBC drivers by using one of the following commands:

earWeblogicDbcp	Builds the WAR file with JDBC drivers. PolicyCenter manages the database connection pool.
-----------------	---

earWeblogicJndi	Builds the WAR file without JDBC drivers. JBoss manages a JNDI database connection pool.
-----------------	--

Procedure

1. Open a command prompt and navigate to the PolicyCenter installation directory.
2. Execute one of the following commands:

```
gwb earWeblogicDbcp
```

```
gwb earWebLogicJndi
```

Both commands generate the PolicyCenter WAR file in the PolicyCenter `dist/war` directory.

Next steps

After completing this procedure, proceed to “Install the PolicyCenter EAR file on WebLogic” on page 87.

Install the PolicyCenter EAR file on WebLogic

Before you begin

Before proceeding, complete “Generate the PolicyCenter EAR file for WebLogic” on page 86.

Procedure

1. Do one of the following:
 - 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`.
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**.

Next steps

After completing this procedure, proceed to “Ways to start PolicyCenter on WebLogic” on page 87.

Ways to start PolicyCenter on WebLogic

It is possible to configure WebLogic to start applications automatically at startup or at restart. In this way, if the WebLogic server ever goes down, it is possible to start Guidewire PolicyCenter automatically.

If you do not configure WebLogic to automatically start applications, then you need to start PolicyCenter manually. See “Start PolicyCenter on WebLogic” on page 110 for details.

Installing PolicyCenter on WebSphere in a production environment

To use WebSphere as the PolicyCenter application server, you create and deploy a WebSphere-specific EAR package file to that application. Generating the EAR file is a multistep process. See the following:

1. “Add a welcome-file-list element to file web.xml” on page 88
2. “Generate the PolicyCenter EAR file for WebSphere” on page 89
3. “Install the PolicyCenter EAR file on WebSphere” on page 89

Important caveats

- Guidewire supports the JDBC drivers shipped with PolicyCenter only for use with externally managed data sources. Guidewire does not support any default JDBC drivers installed with the application server.
- It is not possible to access Guidewire PolicyCenter locally on WebSphere by using the following standard URL:

```
http://localhost:9080/pc/
```

Instead, use the following URL:

```
http://localhost:9080/pc/Start.do
```

Add a welcome-file-list element to file web.xml

Before you begin

Before proceeding, review “Installing PolicyCenter on WebSphere in a production environment” on page 88.

About this task

To ensure that it is possible for WebSphere to access PolicyCenter start file `index.html`, add a `welcome-file-list` element to file `web.xml`.

Procedure

1. Launch Guidewire Studio:
 - a. Open a command prompt and navigate to the PolicyCenter installation directory.
 - b. Execute the following command:


```
gwb Studio
```
2. In the PolicyCenter **Project** window, expand **configuration→deploy→WEB-INF**:
 - a. Open `web.xml`.
 - b. At the bottom of the file, just before the `</web-app>` tag, add the following text:

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

- c. Add servlet definitions as needed.
Review the defined servlets for an example.
 - d. Add a servlet-mapping definition for each servlet that you add.
Review the defined servlets for an example.
3. Save your changes and close `web.xml`.

Next steps

After completing this procedure, proceed to “Generate the PolicyCenter EAR file for WebSphere” on page 89.

Generate the PolicyCenter EAR file for WebSphere

Before you begin

Before proceeding, complete “Add a welcome-file-list element to file web.xml” on page 88.

About this task

You can build the PolicyCenter EAR file for WebSphere with or without JDBC drivers using the following commands:

<code>earWeblogicDbcp</code>	Builds the WAR file with JDBC drivers. PolicyCenter manages the database connection pool.
<code>earWeblogicJndi</code>	Builds the WAR file without JDBC drivers. JBoss manages a JNDI database connection pool.

Procedure

1. Open a command prompt and navigate to the PolicyCenter installation directory.
2. Execute one of the following commands:

```
gwb earWeblogicDbcp
gwb earWeblogicJndi
```

Result

Executing either of these commands does the following:

- It builds the EAR file and place it the PolicyCenter `dist/ear` directory.
- It packages file `modules/configuration/deploy/WEB-INF/web.xml` in the EAR file.

Next steps

After completing this procedure, proceed to “Install the PolicyCenter EAR file on WebSphere” on page 89.

Install the PolicyCenter EAR file on WebSphere

Before you begin

Before proceeding, complete “Generate the PolicyCenter EAR file for WebSphere” on page 89.

Procedure

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 installation `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** screen, verify that your targeted server or cluster is selected.

12. Click **Next**.
13. On the **Metadata for modules** screen, click **Next**.
14. On the **Summary** screen, 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. (Optional) If using JNDI, remove the JDBC JAR files (such as `ojdbc-<version>.jar` or `mssqljdbc-<version>.jar`) from the exploded deployment in the following directory:

```
<WAS Profile>/installedApps/DefaultCell/pc.ear/pc.war/WEB-INF/lib
```

Next steps

After completing this procedure, review the post-installation tasks in “Additional PolicyCenter setup tasks” on page 91. Then, proceed to “Start PolicyCenter on WebSphere” on page 110.

Additional PolicyCenter setup tasks

This topic describes optional PolicyCenter setup tasks. You perform these tasks after you complete the initial installation of your PolicyCenter development or production environment and deploy PolicyCenter to your application server.

Additional installation information

The following topics link to sources for more information.

[Integrating PolicyCenter with ContactManager](#)

To integrate PolicyCenter with ContactManager, see the *Guidewire 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 the *System Administration Guide*.

Change the Super User (su) password

[About this task](#)

In the base configuration, PolicyCenter defines an unrestricted user with first name Super and last name User with a username of su . The default password for this unrestricted user is gw. Guidewire strongly recommends that after you start PolicyCenter for the first time, log into PolicyCenter as user su and change this password.

Note: To change which user is the unrestricted user, see the *System Administration Guide*.

[Procedure](#)

1. Ensure that the PolicyCenter application server is running.
2. Open a browser window.
3. Set the URL to the following:

```
http://server:port/pc/
```

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

```
http://localhost:8180/pc/PolicyCenter.do
```

4. Log into PolicyCenter as user su with password gw.
5. Click the **Preferences** link on the **Desktop** and change the password for user su.

Generate Java API libraries

About this task

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

Guidewire recommends that you generate these APIs after your first initial PolicyCenter installation. Because the toolkit contains the PolicyCenter APIs, regenerate the toolkit after every data model change as well.

Procedure

1. Ensure that the PolicyCenter application server is running.
2. From a command prompt, navigate to the PolicyCenter installation directory.
3. Execute the following command:

```
gwb genJavaApi
```

This command generates the `java-api` directory within the top-level PolicyCenter directory.

Integrating ClaimCenter and PolicyCenter

You can integrate ClaimCenter and PolicyCenter to support claim search from PolicyCenter, policy search from ClaimCenter, and large loss notification from ClaimCenter to PolicyCenter.

Configure ClaimCenter to retrieve policy information

You can configure ClaimCenter to retrieve policy information from PolicyCenter. In ClaimCenter, you register a plugin implementation class that uses a PolicyCenter web service to retrieve policy information for a claim.

Before you begin

Ensure that the PolicyCenter application server is running.

About this task

Note: These instructions describe how to set up ClaimCenter 10 to integrate with either PolicyCenter 9 or PolicyCenter 10. If you are integrating ClaimCenter 10 with PolicyCenter 9, use the instructions in the PolicyCenter 9 documentation for the PolicyCenter side of the integration.

Procedure

1. Open Guidewire Studio™ for ClaimCenter.
2. Navigate in the **Project** window to **configuration**→**config**→**Plugins**→**registry**.
3. Open `IPolicySearchAdapter.gwp`.
4. Click **Remove**  to remove the demonstration implementation, `gw.plugin.policy.impl.PolicySearchPluginDemoImpl`.
5. Click **Add**  and select **Add Gosu Plugin**.
6. Enter one of the following for the **Gosu Class** field, depending on the version of PolicyCenter that you are integrating with ClaimCenter:
 - `gw.plugin.pcentegration.pc1000.PolicySearchPCPlugin` for PolicyCenter 10
 - `gw.plugin.pcentegration.pc900.PolicySearchPCPlugin` for PolicyCenter 9

7. Navigate in the **Project** window to **configuration→config→suite**.
8. Open `suite-config.xml`.
9. Remove the comment markers `<!--` and `-->` from the line for the PolicyCenter URL:

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

10. Update the PolicyCenter URL to match your server and port.
11. Navigate in the **Project** window to **configuration→config** and open `config.xml`.
12. Find configuration parameter `PolicySystemURL`.
13. Set the value of this parameter to the PolicyCenter application URL.

For example, add the following line to this file:

```
<param name="PolicySystemURL" value="http://localhost:8180/pc"/>
```

14. Navigate in the **Project** window to **configuration→gsrsrc** and then to `wsi.remote.gw.webservice.pc`.
15. Open one of the following web service files, depending on the version of PolicyCenter that you are integrating with ClaimCenter:
 - `pc1000.wsc` for PolicyCenter 10
 - `pc900.wsc` for PolicyCenter 9

Guidewire defines the `${pc}` variable for each of the defined web services in `suite-config.xml`.

16. Select all the web services in **Resources** and click **Fetch** .

You must refresh the web services, even if you have made no changes to them.

17. Select the **Settings** tab and review the contents of the **Configuration Provider** class `wsi.remote.gw.webservice.pc.PCConfigurationProvider`.

This class defines the user name and password that ClaimCenter uses to connect with PolicyCenter.

IMPORTANT Guidewire strongly recommends that you change the user name and password from the defaults, `su` and `gw`, to new values.

18. To test the integration, restart the ClaimCenter server.
19. Open the **New Claim** wizard.
20. Search for a Personal Auto or Workers' Compensation policy.

Next steps

“Configuring ClaimCenter to convert PolicyCenter objects” on page 93

Configuring ClaimCenter to convert PolicyCenter objects

You can configure the conversion of the objects that PolicyCenter returns to ClaimCenter in file `pc-to-cc-data-mapping.xml`.

Guidewire Studio™ for ClaimCenter provides different versions of this file for the version of PolicyCenter that ClaimCenter is integrating with.

- To access the ClaimCenter `pc-to-cc-data-mapping.xml` file for PolicyCenter 10, navigate in the **Project** window to **configuration→config→datamapping→pc→1000**.
- To access the ClaimCenter `pc-to-cc-data-mapping.xml` file for PolicyCenter 9, navigate in the **Project** window to **configuration→config→datamapping→pc→900**.

The only object that ClaimCenter sends to PolicyCenter is a `PCClaimSearchCriteria` object. The ClaimCenter search criteria create and populate this object through Gosu code. Any configuration that you do for this object must be done in Gosu code.

The plugin implementation code does not pick up changes to the `pc-to-cc-data-mapping.xml` file automatically. After changing the file, restart the ClaimCenter server to propagate your changes.

See also

- “Configure PolicyCenter to retrieve claim information” on page 94
- “Configure ClaimCenter to retrieve policy information” on page 92

Configure PolicyCenter to retrieve claim information

You can configure PolicyCenter to retrieve claim information from ClaimCenter. In PolicyCenter, you register a plugin implementation class that uses a ClaimCenter web service to retrieve claim information for a policy or account.

Before you begin

Complete the ClaimCenter side of the configuration, described at “Configure ClaimCenter to retrieve policy information” on page 92. Additionally, before starting this procedure, ensure that the ClaimCenter application server is running.

About this task

These instructions describe how to set up PolicyCenter 10 to integrate with either ClaimCenter 10 or ClaimCenter 9. If you are integrating PolicyCenter 10 with ClaimCenter 9, use the instructions in the ClaimCenter 9 documentation for the ClaimCenter side of the integration.

Procedure

1. Open Guidewire Studio™ for PolicyCenter.
2. Navigate in the **Project** window to **configuration→config→suite**.
3. Open file `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. Navigate in the **Project** window to **configuration→gsrsrc** and then to `wsi.remote.gw.webservice.cc`.
7. Open one of the following web service files, depending on the version of ClaimCenter that you are integrating with PolicyCenter:
 - `cc1000.wsc` for ClaimCenter 10
 - `cc900.wsc` for ClaimCenter 9

File `suite-config.xml` defines the `${cc}` variable for the web service.

8. Ensure that ClaimCenter is running, and then select the web service in **Resources** and click **Fetch** . You must refresh the web service, even if you have made no changes to it.
9. Select the **Settings** tab and review the contents of the listed **Configuration Provider** class.

This class defines the user name and password that PolicyCenter uses to connect with ClaimCenter.

IMPORTANT Guidewire strongly recommends that you change the user name and password from the defaults, `su` and `gw`, to new values.

10. Navigate in the **Project** window to **configuration→config→Plugins→registry**.
11. Open `IClaimSearchPlugin.gwp`.
12. Click **Remove Plugin**  to remove the following demonstration implementation:


```
gw.plugin.claimsearch.impl.GWDemoClaimSearchPlugin
```
13. Click **Add Plugin**  and select **Add Gosu Plugin**.

14. 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:
 - `gw.plugin.claimsearch.cc1000.GWClaimSearchPlugin` for ClaimCenter 10
 - `gw.plugin.claimsearch.cc900.GWClaimSearchPlugin` for ClaimCenter 9
15. Navigate in the **Project** window to **configuration**→**config** and open `config.xml`.
16. Set the `ClaimSystemURL` parameter to the ClaimCenter URL.
For example, the base configuration URL in a local ClaimCenter installation is the following:

```
http://localhost:8080/cc
```
17. Restart the PolicyCenter server to pick up these changes.

Next steps

“Defining authentication between PolicyCenter and ClaimCenter” on page 95

Defining authentication between PolicyCenter and ClaimCenter

PolicyCenter and ClaimCenter define the user name and password used to communicate with the other application in configuration provider classes.

- PolicyCenter provides the following class that defines the user name and password that PolicyCenter uses to connect with ClaimCenter:

```
wsi.remote.gw.webservice.cc.CCConfigurationProvider
```

In the base configuration, PolicyCenter defines the user name `su` and password `gw` in this class.

- ClaimCenter provides the following class that defines the user name and password that ClaimCenter uses to connect with PolicyCenter:

```
wsi.remote.gw.webservice.pc.PCConfigurationProvider
```

In the base configuration, ClaimCenter defines the user name `su` and password `gw` in this class.

IMPORTANT Guidewire strongly recommends that, as a matter of security, you change the user name and password that each application defines. You must also supply a corresponding user in the other application. That new user must have the `soapadmin` permission, and, at a minimum, the permissions needed to view policies and accounts .

For example:

1. In Studio for PolicyCenter, define a user name and password in `CCConfigurationProvider`.
2. Log in to the ClaimCenter server and define a user that has the same user name and password that you specify in `CCConfigurationProvider`.
3. Assign the user a role that has the `soapadmin` permission, and, at a minimum, the permissions needed to view policies and accounts.
4. In Studio for ClaimCenter, define a user name and password in `PCConfigurationProvider`.
5. Log in to the PolicyCenter server and define a user that has the same user name and password that you specify in `PCConfigurationProvider`.
6. Assign the user a role that has the `soapadmin` permission, and, at a minimum, the permissions needed to view policies and accounts.

See also

- *Guidewire Contact Management Guide*

Configure ClaimCenter to send large loss notification to PolicyCenter

You can configure ClaimCenter to send PolicyCenter a large loss notification whenever the value of the Gross Total Incurred amount on a claim exceeds a defined threshold.

Before you begin

- “Configure ClaimCenter to retrieve policy information” on page 92
- “Configure PolicyCenter to retrieve claim information” on page 94

About this task

Ensure that the PolicyCenter application server is running. It is not necessary for the ClaimCenter server to be running.

Procedure

1. Open Guidewire Studio™ for ClaimCenter.
2. Navigate in the **Project** window to **configuration→config→Rule Sets** and then:
 - a. Navigate to **Preupdate→TransactionSetPreupdate** and select the check box that enables the rule TPU05000 - Large Loss Notification.
 - b. Navigate to **EventMessage→EventFired** and select the check box that enables the rule EFR06000 - Policy System Notification.
3. Expand **configuration→config→Messaging**.
4. Open messaging-config.xml in the Messaging editor.
5. Select the following messaging destination:

```
ID-69, Name=Java.MessageDestination.PolicySystemNotification.Name
```

6. Clear the **Disable destination** check box at the top right of the editor to enable this message destination.
7. In the **Project** window, expand **configuration→config→Plugins→registry**.
8. Open policySystemNotificationTransport.gwp in the plugin editor.
9. Clear the **Disabled** check box under the **Server:** field.
10. Open IPolicySystemNotificationPlugin.gwp.
The editor for this plugin opens with the default plugin implementation selected.
11. Remove the demonstration implementation by clicking the **Remove Plugin**  icon.
12. Click **Add Plugin**  and select **Add Gosu Plugin**.
13. Enter the following for the **Gosu Class** field:

```
gw.plugin.policy.notification.pc1000.PCPolicySystemNotificationPlugin
```

14. If you have not done so already, update suite-config.xml to enable ClaimCenter to work with PolicyCenter.
 - a. Navigate in the **Project** window to **configuration→config→suite**.
 - b. Open suite-config.xml.
 - c. If present, remove the comment markers `<!--` and `-->` from the line defining the PolicyCenter URL:

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

- d. If necessary, change the URL for the PolicyCenter server to the URL used in your configuration.
15. If you have not done so already, set the PolicyCenter URL in the PolicySystemURL configuration parameter:
 - a. In the Studio **Project** window, expand **configuration→config**.
 - b. Open config.xml.
 - c. 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"/>
```

16. Navigate in the **Project** window to **configuration→gsrsrc** and then to `wsi.remote.gw.webservice.pc`.

17. Double-click `pc1000.wsc` to open this web service collection in the editor.
18. Select the following web service in **Resources**:
`${pc}/ws/gw/webService/pc/pc1000/ccintegration/ClaimToPolicySystemNotificationAPI?wsdl`
File `suite-config.xml` defines the `${pc}` variable for the web service.
19. Ensure that the PolicyCenter server is running, and then click **Fetch**  to refresh the web service.
You must refresh the web service, even if you have made no changes to it.
20. Click the **Settings** tab and review the contents of the **Configuration Provider** class
`wsi.remote.gw.webService.pc.PCConfigurationProvider`.
This class defines the user name and password that ClaimCenter uses to connect with PolicyCenter.

IMPORTANT Guidewire strongly recommends that you change these values from their default `su` and `gw` values.

21. Restart the ClaimCenter server so these changes take effect.

Next steps

“Test ClaimCenter large loss notification integration with PolicyCenter” on page 97

Test ClaimCenter large loss notification integration with PolicyCenter

Before you begin

“Configure ClaimCenter to send large loss notification to PolicyCenter” on page 95

About this task

In a development system, it is possible to test the large loss notification integration between ClaimCenter 10 and PolicyCenter 10.

Procedure

1. Log into Guidewire ClaimCenter.
2. Add a claim in ClaimCenter with a loss that exceeds the threshold for that type of claim.
3. Log into Guidewire PolicyCenter.
4. Review the activities for the policy associated with the claim and verify that ClaimCenter successfully created a large loss notification.

Result

If the integration works correctly, PolicyCenter adds a referral reason and an activity for that policy.

Integrating BillingCenter and PolicyCenter

The default installations of both PolicyCenter and BillingCenter support integration between the two applications. With this integration, PolicyCenter and BillingCenter can exchange information about the following:

- Accounts
- Billing
- Policies
- Producers
- Producer codes

After integrating the two applications:

- PolicyCenter payment screens shows payment plans retrieved from BillingCenter.
- PolicyCenter transmits your payment plan selection to BillingCenter and saves the information with the policy period.
- BillingCenter and PolicyCenter share account and policy period information.
- BillingCenter sends delinquency notices to PolicyCenter.
- PolicyCenter screens show links that you can click to view data in BillingCenter.

Important BillingCenter-PolicyCenter integration caveat

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

BillingCenter integration points with PolicyCenter

In the base configuration, BillingCenter provides a default implementation of the `IPolicySystemPlugin` plugin that interfaces with specialized PolicyCenter web services that Guidewire provides with PolicyCenter 10 and PolicyCenter 9.

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

- `gw.webservice.policycenter.bc1000.BillingAPI`
- `gw.webservice.policycenter.bc1000.BillingSummaryAPI`
- `gw.webservice.bc.bc1000.BCAPI`
- `gw.webservice.bc.bc1000.PaymentInstrumentAPI`
- `gw.webservice.policycenter.bc900.BillingAPI`
- `gw.webservice.policycenter.bc900.BillingSummaryAPI`
- `gw.webservice.bc.bc900.BCAPI`
- `gw.webservice.bc.bc900.PaymentInstrumentAPI`

PolicyCenter integration points with BillingCenter

In the base configuration, PolicyCenter provides default implementations of the plugin interfaces `IBillingSummaryPlugin` and `IBillingSystemPlugin`. These plugin implementation classes use BillingCenter web services to send information to and retrieve information from BillingCenter.

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

- `gw.webservice.pc.pc1000.job.PolicyRenewalAPI`
- `gw.webservice.pc.pc1000.job.CancellationAPI`
- `gw.webservice.pc.pc900.job.PolicyRenewalAPI`
- `gw.webservice.pc.pc900.job.CancellationAPI`

See also

- *Application Guide*
- *Integration Guide*

Enable the policy system plugin in BillingCenter

Before you begin

Before starting this procedure, ensure that the PolicyCenter application server is running. It is not necessary for the BillingCenter server to be running.

Procedure

1. Open Guidewire Studio™ for BillingCenter.
2. Navigate in the **Project** window to **configuration**→**config**→**Plugins**→**registry**.
3. Open `IPolicySystemPlugin.gwp` in the Plugin editor.
4. Set **Gosu Class** to `gw.plugin.pas.pc1000.PCPolicySystemPlugin`.
5. Navigate in the **Project** window to **configuration**→**config**→**suite**.
6. Double-click `suite-config.xml` to open the file.
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. Navigate in the **Project** window to **configuration**→**gsrsrc** and then to `wsi.remote.gw.webservice.pc`.
10. Open one of the following web service collection files, depending on the version of PolicyCenter that you are integrating with BillingCenter:
 - `pc1000.wsc` for PolicyCenter 10
 - `pc900.wsc` for PolicyCenter 9

The web service definitions in this file use the `${pc}` variable defined in `suite-config.xml`.

11. Ensure that the PolicyCenter server is running, and then select all the web services in **Resources** and click **Fetch** .

You must refresh the web services, even if you have made no changes to them.

12. Select the **Settings** tab and review the contents of the **Configuration Provider** class.

This class defines the user name and password that BillingCenter uses to connect with PolicyCenter.

IMPORTANT Guidewire strongly recommends that you change these values from their default values.

13. Stop the PolicyCenter application server.

Next steps

“Enable the billing summary plugin in PolicyCenter” on page 99

Enable the billing summary plugin in PolicyCenter

Before you begin

“Enable the policy system plugin in BillingCenter” on page 98

About this task

Before starting this procedure, ensure that the BillingCenter application server is running. It is not necessary for the PolicyCenter server to be running.

Procedure

1. Open Guidewire Studio™ for PolicyCenter.
2. Navigate in the **Project** window to **configuration**→**config**→**Plugins**→**registry** and open `IBillingSummaryPlugin.gwp` in the Plugin editor.
3. Change **Gosu Class** to one of the following, depending on the version of BillingCenter that you are integrating with:
 - `gw.plugin.billing.bc1000.BCBillingSummaryPlugin` for BillingCenter 10
 - `gw.plugin.billing.bc900.BCBillingSummaryPlugin` for BillingCenter 9

4. In the Studio **Project** window folder, open `IBillingSystemPlugin.gwp`.
5. Change **Gosu Class** to one of the following plugin implementation classes, depending on the version of BillingCenter that you are integrating with:
 - `gw.plugin.billing.bc1000.BCBillingSystemPlugin` for BillingCenter 10
 - `gw.plugin.billing.bc900.BCBillingSystemPlugin` for BillingCenter 9
6. Navigate in the **Project** window to **configuration**→**config** and open `config.xml`.
7. Set the value of the `BillingSystemURL` parameter to the BillingCenter URL.

For example, in the base configuration, the URL for a local BillingCenter installation is the following:

```
http://localhost:8580/bc/BillingCenter.do
```

Configuration parameter `BillingSystemURL` defines the URL for an exit point. Guidewire configures exit points 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 the *Integration Guide*.

8. If archiving is enabled in BillingCenter, set the following PolicyCenter configuration parameters:
 - a. Set `BillingSystemArchiveEnabled` to `true`.
 - b. Set the value of `BillingSystemArchivePolicyPeriodDays` to a positive value indicating the minimum number of days after the term end date before the billing policy period is archived.
 This value must be less than or equal to the setting in the billing system. Guidewire recommends setting the value to be equal and keeping the value of `BillingSystemArchivePolicyPeriodDays` synchronized with the value of `ArchivePolicyPeriodDays` in BillingCenter.
9. Navigate in the **Project** window to **configuration**→**config**→**suite** and 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. Navigate in the **Project** window to **configuration**→**gsrsc** and then to `wsi.remote.gw.webservice.bc`.
13. Open one of the following web service collection files, depending on the version of BillingCenter that you are integrating with PolicyCenter:
 - `bc1000.wsc` for BillingCenter 10
 - `bc900.wsc` for BillingCenter 9

The web service definitions in this file use the `${bc}` variable defined in `suite-config.xml`.

14. Select all the web services in **Resources** and click **Fetch** .

You must refresh the web services, even if you have made no changes to them.

15. Click the **Settings** tab and review the contents of the **Configuration Provider** class.

This class defines the user name and password that PolicyCenter uses to connect with BillingCenter.

IMPORTANT Guidewire strongly recommends that you change these values from their default values.

16. Stop the BillingCenter application server.

Next steps

“Defining authentication between PolicyCenter and BillingCenter ” on page 101

Defining authentication between PolicyCenter and BillingCenter

PolicyCenter and BillingCenter define the user name and password used to communicate with the other application in configuration provider classes.

- PolicyCenter provides the following class that defines the user name and password that PolicyCenter uses to connect with BillingCenter:

```
wsi.remote.gw.webservice.bc.BCConfigurationProvider
```

In the base configuration, PolicyCenter defines the user name `pu` and password `gw` in this class.

- BillingCenter provides the following class that defines the user name and password that BillingCenter uses to connect with PolicyCenter:

```
wsi.remote.gw.webservice.pc.PCConfigurationProvider
```

In the base configuration, BillingCenter defines the user name `su` and password `gw` in this class.

IMPORTANT Guidewire strongly recommends that, as a matter of security, you change the user name and password that each application defines. You must also supply a corresponding user in the other application. That new user must have the `soapadmin` permission, and, at a minimum, the permissions needed to view policies and accounts.

For example:

1. In Studio for PolicyCenter, define a user name and password in `BCConfigurationProvider`.
2. Log in to the BillingCenter server and define a user that has the same user name and password that you specify in `BCConfigurationProvider`.
3. Assign the user a role that has the `soapadmin` permission, and, at a minimum, the permissions needed to view policies and accounts.
4. In Studio for BillingCenter, define a user name and password in `PCConfigurationProvider`.
5. Log in to the PolicyCenter server and define a user that has the same user name and password that you specify in `PCConfigurationProvider`.
6. Assign the user a role that has the `soapadmin` permission, and, at a minimum, the permissions needed to view policies and accounts.

See also

- *Guidewire Contact Management Guide*

Start BillingCenter in an integrated environment

Before you begin

Set up authentication for the integration as described in “Defining authentication between PolicyCenter and BillingCenter ” on page 101

About this task

Start Guidewire BillingCenter before starting PolicyCenter. After you enable the integration between PolicyCenter and BillingCenter, PolicyCenter sends `Producer`, `ProducerCode`, `Account`, and `Policy` entity instances to BillingCenter as the application starts. Therefore, you must start BillingCenter before PolicyCenter, so that BillingCenter is available to receive this data.

Note: If you intend to load sample data into PolicyCenter, be sure to load sample data into BillingCenter first.

Procedure

1. Start the BillingCenter application server:
 - a. Open a command prompt in the BillingCenter installation directory.

- b. Execute the following command:


```
gwb runServer
```
2. After BillingCenter fully starts, open a supported web browser and enter the URL to BillingCenter. For example, in the default configuration, enter the following URL:


```
http://localhost:8580/bc/BillingCenter.do
```
3. For testing, development, or demonstration purposes, load sample data. If PolicyCenter has sample data, you must import sample data for BillingCenter for the integration to work properly.
 - a. Log in to BillingCenter as a user with administrative privileges.
 - b. Press Alt+Shift+T.
 - c. Click the **Internal Tools** tab.
 - d. Click **BC Sample Data** in the Sidebar, and then click **Import Sample Data**.
 - e. After the data set loads, click the **Options** menu , and then click **Return to BillingCenter**.

Next steps

“Start PolicyCenter in an integrated environment” on page 102

Start PolicyCenter in an integrated environment

Before you begin

“Start BillingCenter in an integrated environment” on page 101

About this task

After you enable the integration between PolicyCenter and BillingCenter, PolicyCenter sends Producer, ProducerCode, Account, and Policy entity instances to BillingCenter as the application starts. Therefore, you must start BillingCenter before PolicyCenter, so that BillingCenter is available to receive this data.

Note: If you load sample data into PolicyCenter, be sure to load sample data into BillingCenter first.

Procedure

1. Start the PolicyCenter application server:
 - a. Open a command prompt in the PolicyCenter installation directory.
 - b. Execute the following command:


```
gwb runServer
```
2. After PolicyCenter fully starts, open a supported web browser and enter the URL to PolicyCenter. For example, in the default configuration, enter the following URL:


```
http://localhost:8180/pc/PolicyCenter.do
```
3. For testing, development or demonstration purposes, load sample data.
 - a. Log into PolicyCenter as user with administrative privileges.
 - b. Press Alt+Shift+T.
 - c. Open 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, click the **Options** menu , and then click **Return to PolicyCenter**.

Next steps

“Verify the integration between PolicyCenter and BillingCenter” on page 103

Verify the integration between PolicyCenter and BillingCenter

Before you begin

“Start PolicyCenter in an integrated environment” on page 102

About this task

After you enable the integration between PolicyCenter and BillingCenter, PolicyCenter sends **Producer**, **ProducerCode**, **Account**, and **Policy** entity instances to BillingCenter as the server starts. It is important that you verify that BillingCenter receives these entity instances.

Procedure

1. Verify that PolicyCenter transferred policy information to BillingCenter.
 - a. In PolicyCenter, navigate to the **Search** tab **Policies** and click **Policies**.
 - b. In the **Search Policies** screen, enter enough data to search for a policy.
For example, if using the small sample data set, 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 PolicyCenter transferred account information 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 then paste the account number into 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 PolicyCenter transferred producer and producer code information 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.
4. In the **Name** column, click the link to the producer to see the producer **Summary** page.

Archiving in a production environment

After you enable PolicyCenter archiving, you cannot disable archiving entirely unless you drop the application database. Additionally, after enabling archiving, even if you then disable archiving, you still see references to Policy Term archiving.

Do not attempt to disable archiving in a production environment after you have enabled this functionality.

WARNING Do not attempt to reset the value of `ArchiveEnabled` from `true` to `false`. After you set `ArchiveEnabled` to `true` and start the server, you cannot change the value of this parameter again. If you reset the value to `false`, the server does not start. Also, do not remove the archive work queue. Removing this work queue after enabling archiving prevents that application server from starting.

Implementing policy term archiving

If you intend to implement Policy Term archiving in your production PolicyCenter environment, first test the end-to-end archiving process in a development environment.

Disabling policy term archiving

If you do not intend to implement Policy Term archiving in your production PolicyCenter environment, do the following:

- Accept the default archiving configuration, which Guidewire disables by default.
- Manually disable all unused archiving configuration elements.

See also

- “Archiving in a development environment” on page 58
- “Enable archiving in Guidewire PolicyCenter” on page 59
- “Disabling Guidewire PolicyCenter archiving” on page 60
- *Application Guide*

About PolicyCenter Rating Management

Guidewire Rating Management is available within PolicyCenter. However, Guidewire licenses Rating Management 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 enabling Rating Management, you must enable the rating plugin that works with Rating Management.

See also

- *Application Guide*
- *Integration Guide*

Disable reinsurance ceding work queue

About this task

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

Procedure

1. In Studio, expand **configuration**→**config**→**workqueue**.

- a. Double-click `work-queue.xml` to open it.
- b. Search for and comment out the following block by adding `<!--` before the block and `-->` after it the code block.

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

2. Save your changes.

Connecting a web client to PolicyCenter

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:/8180/pc/PolicyCenter.do`

`http://AppServer2:/8280/ab/ContactManager.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	• 8080 – ClaimCenter • 8180 – PolicyCenter • 8280 – ContactManager • 8580 – BillingCenter
JBoss	8080
Tomcat	8080
WebLogic	7001
WebSphere	9080

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

See also

- See “Web client information” on page 47 for a list of required software and hardware for client computers accessing PolicyCenter.

Configure Microsoft Windows accessibility for Firefox

About this task

If a user clicks a cell in an editable list view, the dotted border around action 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 uses a Windows accessibility setting to determine this behavior.

Procedure

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.

About single sign-on authentication

It is possible to configure PolicyCenter to use single sign-on (SSO) authentication. In single sign-on operation, as the user logs into PolicyCenter:

- PolicyCenter generates a unique Cross-Site Request Forgery (CSRF) token for the user session.
- PolicyCenter forwards user information, including the CSRF token, to the authentication provider.
- The authentication provider checks the credentials for the user.
- The authentication provider returns an authentication confirmation back to PolicyCenter if the credentials are valid.

PolicyCenter includes the CSRF token in each authentication request and uses the CSRF token to verify the legitimacy of the user request.

See also

- *Integration Guide*

Configure single sign-on authentication

About this task

Guidewire provides the following configuration as a basic example. 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.

Procedure

1. Create custom Gosu class CustomAuthServlet.gs:
 - a. Open Guidewire Studio™ for PolicyCenter.
 - b. In the Studio **Project** window, expand **configuration**→**gsrsrc**.
 - c. Right-click **gsrsrc** and click **New**→**Package**.
 - d. Enter a package name for upgrade purposes.
For example, enter something such as *companyName.auth*.
 - e. Right-click the newly created 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. Add your custom servlet to the list of valid PolicyCenter servlets:

- a. Expand **configuration**→**config**→**servlets**.
- b. Open **servlets.xml**.
- c. Add the name of your custom servlet to the list.

For example:

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

3. Create custom Gosu class `AuthServicePlugin.gsu` and place the class in your custom authentication package. For example:

```

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;
    }
}

```

4. Create custom Gosu class `AuthSourceCreator.gs` and place the class in your custom authentication package. For example:

```

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.

5. Associate your custom `AuthServicePlugin` class with the `AuthenticationServicePlugin` plugin.
 - a. Expand **configuration**→**config**→**Plugins**→**registry**:
 - b. Open `AuthenticationServicePlugin.gwp`.
 - c. Click **Remove Plugin**  to remove the default plugin.
 - d. Click **Add Plugin**  and select **Add Gosu Plugin**.
 - e. For **Gosu Class**, enter the `AuthServicePlugin.gs` class, including the fully qualified package.
6. Associate your custom `AuthSourceCreator` class with the `AuthenticationSourceCreatorPlugin` plugin.
 - a. Open `AuthenticationSourceCreatorPlugin.gwp`.
 - b. Click **Remove Plugin**  to remove the default plugin.
 - c. Click **Add Plugin**  and select **Add Gosu Plugin**.
 - d. For **Gosu Class**, enter the `AuthSourceCreator.gs` class, including the fully qualified package.
7. Create an entry point for the PolicyCenter 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.
8. 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.

9. Create a BaseURLBuilderPlugin plugin implementation:
 - 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 **Add Plugin**  and select **Add Java Plugin**.
 - e. Enter com.guidewire.pl.web.render.html.BaseURLBuilderImpl for the **Java Class**.
10. Test your work:
 - a. Create a test HTML page on your local PolicyCenter server.
 - b. Include the following form on the HTML page:

```
<form name="input" action="http://localhost:8180/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

How you start Guidewire PolicyCenter depends on the specific application server that you use.

See also

- *System Administration Guide*

Start PolicyCenter on JBoss

About this task

After installing PolicyCenter on JBoss, PolicyCenter starts as you start JBoss. To start JBoss, use the run command in the JBoss bin directory. If you execute the run command without any parameters, the command launches JBoss using the default server configuration.

Start PolicyCenter on Tomcat on Windows

About this task

After installing PolicyCenter on Tomcat, PolicyCenter starts as you start Tomcat. To start the Tomcat server on Windows, run the following script from the Tomcat installation directory:

```
bin/startup.bat
```

Start Tomcat in same command prompt

About this task

By default, Tomcat starts up in a new window. If Tomcat encounters startup errors, the new window often closes automatically far too quickly for you to read any error messages in it. To run Tomcat from the same command prompt in which you entered the startup command, perform the following steps:

Procedure

1. Locate the line in the script that runs Tomcat:

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

2. Change the start option in the command to run.

3. Save the script, and then run the script again.

Tomcat then runs in the same command window, which you can use to view any error messages.

Start PolicyCenter on WebLogic

Procedure

1. Start the WebLogic Administration Server if it is not already running.
2. Open the WebLogic Administration Console.
3. Under **Domain Structure**, click **Deployments**.
 - a. Select the check box for PolicyCenter.
 - b. Click **Start**→**Servicing all requests**.
The WebLogic Administration Console informs you of the deployments that you selected to be started.
 - c. Click **Yes**.

4. Navigate to and select the PolicyCenter application.

WebLogic identifies PolicyCenter by the name that you gave to the deployed EAR file.

5. Click **Deploy Application**.

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

Start PolicyCenter on WebSphere

About this task

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

Procedure

1. Open the WebSphere Administrative Console.
2. Click **Applications**→**Application Types**→**WebSphere enterprise applications**.
3. Select the check box next to the PolicyCenter application, abbreviated by default as **pc**.
4. Do one of the following:
 - If PolicyCenter is already running, and you want to restart PolicyCenter, restart WebSphere.
 - If PolicyCenter is not running, click **Start**. It is possible for PolicyCenter to take a few minutes to start.

Tune memory settings for application processes

If a particular process is running slowly, you can increase the amount of memory available to it by tuning a set of memory parameters. It is possible that the additional memory allocation can improve the process performance. Use the following table to identify the parameters that you can change in file `gradle.properties`, in the PolicyCenter installation directory.

Property	Description	Possible values	Default values
<code>custDistGosuCompileMaxHeapSize</code>	Maximum heap size for the JVM that performs Gosu compilation.	The number of gigabytes, followed by the letter g.	16g
<code>custDistGosuCompileMinHeapSize</code>	Minimum heap size for the JVM that performs Gosu compilation.	The number of gigabytes, followed by the letter g.	2g

Property	Description	Possible values	Default values
custDistGosudocMaxHeapSize	Maximum heap size for the JVM that performs Gosudoc generation.	The number of gigabytes, followed by the letter g.	16g
custDistGosudocMinHeapSize	Minimum heap size for the JVM that performs Gosudoc generation.	The number of gigabytes, followed by the letter g.	2g
custDistJavaCompileMaxHeapSize	Maximum heap size for the JVM that performs Java compilation.	The number of gigabytes, followed by the letter g.	16g
custDistJavaCompileMinHeapSize	Minimum heap size for the JVM that performs Java compilation.	The number of gigabytes, followed by the letter g.	4g
custDistJavaCompileSchemaSourcesMaxHeapSize	Maximum heap size for the JVM that performs compilation of XML schema generated source for the task genSchemaJar.	The number of gigabytes, followed by the letter g.	16g
custDistJavaCompileSchemaSourcesMinHeapSize	Minimum heap size for the JVM that performs compilation of XML schema generated source for the task genSchemaJar.	The number of gigabytes, followed by the letter g.	4g
custDistJavaExecMaxHeapSize	Maximum heap size for all of the tasks that run in a separate JVM, such as genDataDictionary.	The number of gigabytes, followed by the letter g.	4g
custDistStudioMaxHeapSize	The maximum amount of memory available to Guidewire Studio.	The number of gigabytes, followed by the letter g.	4g
custDistStudioBuildProcessHeapSize	The initial default value for the Build process heap size setting in the Guidewire Studio Settings dialog. After you run Studio for the first time, PolicyCenter ignores this property.	The number of megabytes.	6000
org.gradle.jvmargs	Minimum and maximum heap sizes for the JVM that runs the remaining tasks, such as codegen.	Minimum: -Xms, followed by the number of gigabytes, followed by the letter g. Maximum: -Xmx, followed by the number of gigabytes, followed by the letter g.	-Xms1g -Xmx4g

Configuring free-text search

This topic describes optional PolicyCenter setup tasks. You perform these tasks after you complete the initial installation of your PolicyCenter development or production environment and deploy PolicyCenter to your application server.

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*.

Overview of free-text search setup

PolicyCenter free-text search depends on a full-text search engine, the Guidewire Solr Extension. In a production environment, Guidewire supports running the Guidewire Solr Extension in a standalone mode. 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](#)

- *Configuration Guide*

Free-text search 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 Java Virtual Machine (JVM) instance rather than in the server 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.

You can configure free-text search for external or embedded operation in development environments installed on a Tomcat application server or on the bundled QuickStart application server. You can configure free-text search only for external operation in development environments installed on other supported application servers. Production environments support only external operation regardless of application server.

Note: In PolicyCenter, you can avoid setting up the free-text batch load process in small-scale development environments. In such a case, use the Sync Policy Index feature. In production deployments, Guidewire strongly recommends that you configure the free-text batch load process.

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. Your only set-up task in embedded operation involves changes to the `solrserver-config.xml` file. This file configures how PolicyCenter works with the Guidewire Solr Extension, including connection information, and whether the mode of operation is external or embedded.

See also

- “Configuring free-text search for embedded operation” on page 117

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 in the following location in the PolicyCenter installation directory:

`solr/pc-gwsolr.zip`

The Guidewire Solr Extension runs self-hosted in a production-grade Jetty container 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.

Guidewire Solr home directory

Before you implement free-text search, you must 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 on which 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:

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

All Guidewire free-text setup instructions assume the use `/opt/gwsolr` for the Guidewire Solr home directory. If you do not install the Guidewire Solr Extension in its default directory, you must perform additional configuration steps.

Free-text batch load command

The Guidewire Solr Extension provides a command-line utility to extract policy data from the PolicyCenter relational database and load the extracted data into the Guidewire Solr Extension.

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 **Free-text Search** page from the **Internal Tools** tab.

The free-text batch load command 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.

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 **Internal 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

- “About the free-text batch load command” on page 119
- *Configuration Guide*
- *System Administration Guide*

Securing database credentials for free-text batch load

Free-text batch load for the production instances of the Guidewire Solr Extension requires the PolicyCenter production database credentials. You must keep these credentials secure. To this end, Guidewire strongly recommends that you load the database credentials that batch load requires from a separate property file. This property file must be stored in a secure location. Alternatively, list the property file in a transient OS environment variable set within the batch load launch script.

The following example contains the batch load database credentials of a property file called `solrpass.properties`:

```
db.user=scott
db.password=tiger
```

Guidewire discourages you from placing this property file or any other file requiring security in the document conf folder or in the Solr home folder. Guidewire Solr Extension uploads files from these locations into ZooKeeper. If the property file is in either of these locations, the security for the file will be compromised.

Instead, create a new folder either under the Solr root folder or the document folder. Consider naming such a new folder `secure`. In addition, use operating system tools to restrict access to the property file.

When addressing property files, adhere to the following guidelines. Refer to the Solr home folder using `${solrRoot}`. Refer to the document folder using `${documentRoot}`. Prefix OS environment variable references with `osenv`.

A new node exists in the `batchload-config.xml` file that allows you to inject the proper credentials file for interacting with Guidewire Solr Extension. The name of the node is `solrCredentials`. In addition, the `batchload-config.xml` file supports a variety of mechanisms for property substitution. This combination of features allows you to avoid hard coding credentials consistent with the Guidewire recommendation:

```
<document>
...
<!-- The property file referenced here defines the solr.username and solr.password properties -->
<propertyFilename="${documentRoot}/secure/solrpass.properties"/>
...
<!-- The solrCredentials node uses the solr.username and solr.password properties -->
<solrCredentials username="${solr.username}" password="${solr.password}"/>
...
</document>
```

The following code example uses the property file for the database credentials in the datasource URL:

```
<datasourcename="ds_orcl"driver="oracle.jdbc.OracleDriver"url="jdbc:oracle:thin:${db.user}/${db.password}/
  gwapp@testhost:1521:orcl"env="test"/>
<datasourcename="ds_orcl"driver="oracle.jdbc.OracleDriver"url="jdbc:oracle:thin:${osenv.DB_USER}/${osenv.DB_PASSWORD}/
  gwapp@testhost:1521:orcl"env="test"/>
```

Securing Guidewire Solr Extension

Securing Guidewire Solr Extension both for the server and the client requires two key configurations. The first is the requirement of basic authentication. The second is the activation of SSL.

Requiring authentication in Guidewire Solr Extension

Guidewire recommends that you secure communications between PolicyCenter and the Solr application. To this end, you can configure both PolicyCenter and the Solr application for basic authentication. Note that if you do configure PolicyCenter or the Solr application for basic authentication, Guidewire recommends that you also configure the respective application to run with SSL activated. Otherwise, basic authentication will not provide effective security.

Configuring PolicyCenter to use basic authentication when communicating with Solr

To configure PolicyCenter to use basic authentication when communicating with Solr, one alternative is to store username and password properties in the `solrserver-config.xml` file. The following code example shows how to store these properties in an HTTP server environment:

```
<solrserver name="solr_instance_name" type="http">
  <param name="host" value="localhost"/>
  <param name="port" value="8983"/>
  <param name="securetransport" value="true"/>
  <param name="username" value="solr_user"/>
  <param name="password" value="the_password"/>
</solrserver>
```

If you are in a cloud server environment, substitute `cloud` for the `type` property assignment as in the following example:

```
<solrserver name="solr_cloud_name" type="cloud">
  <param name="zkhosts" value="localhost:2181/pc"/>
  <param name="username" value="solr_user"/>
  <param name="password" value="the_password"/>
</solrserver>
```

Note: Guidewire does not recommend using this alternative in production for security reasons. Clear text credentials can upload to a ZooKeeper installation in the cloud environment. ZooKeeper is accessible by HTTP and is not secure.

As an alternative to using clear text credentials, use the `Credentials` plugin. If you register this plugin, the Solr server manager inside the Guidewire Solr Extension will request credentials for the SOLR key. If the plugin returns credentials in response to such a request, the credential values will supersede any clear text credentials in the `solrserver-config.xml` file.

Configuring the Solr application for basic authentication

To configure the Solr application to use basic authentication, you must define credentials for it. Defining credentials for the Solr application requires producing a credentials hash. To create this hash, use the commands called `createuser.bat` and `createuser.sh` for a given username and password pair. These commands are in the `/opt/gwsolr/pc/bin` folder by default. An example of how to use a `createuser` command is as follows:

```
C:\opt\gwsolr\pc\bin>.\createuser
Usage: createuser -user username [-password password] [-solrhome solrhome_folder] [-debug]
```

In the `createuser` command, if you enter a username without a password, you will receive a prompt to create a password. Otherwise, you will receive the following instruction:

```
Add the following entry to the authentication section of $SOLR_HOME/security.json.
"credentials": {
  "solr": "LkbT+UV72d1Wq8zT/e9hyGFIIZSocHwQtK/h1P/4N2g= gN6KFzYN1Iv0wgIyQvHUx+yX8phzY60ivyRGnpyp2to="
}
Be sure to enable the SOLR_AUTH* properties in bin/solr.in.cmd or bin/solr.in.sh.
```

If you include the `solrhome` option in the `createuser` command and no `security.json` file exists at the `solrhome` location, the file will be created with default content. This default content will include the generated password hash.

See also

- https://lucene.apache.org/solr/guide/6_6/basic-authentication-plugin.html#basic-authentication-plugin

Configuring Guidewire Solr Extension to run with SSL activated

You can configure both the Guidewire Solr Extension server and Guidewire Solr Extension clients to run with SSL activated. Note that if you do configure the server or clients in this way, Guidewire recommends that you also configure the respective server or clients for basic authentication. Otherwise, running the respective server or clients with SSL activated will not provide effective security.

Configuring Guidewire Solr Extension server to run with SSL activated

Single HTTP server configuration

In a single HTTP server configuration, set the `securetransport` property to `true` in the corresponding `solrserver` element of the `solrserver-config.xml` file. The following code example shows this configuration:

```
<solrserver name="solr_ssltest_instance" type="solrhost">
  <param name="host" value="localhost"/>
  <param name="port" value="8983"/>
  <param name="securetransport" value="true"/>
</solrserver>
```

In addition, configure the appropriate certificates in accordance with instructions for the underlying Java Virtual Machine (JVM) or application server.

Cloud server configuration

In a cloud server configuration, set the security flag that enables Guidewire Solr Extension instances for secure transport. To set the flag, run the `gwzkccli - zkhost` command as the following Windows example shows:

```
gwzkccli -zkhostlocalhost:2181 -cmd put /pc/clusterprops.json {"urlScheme":"https"}
```

Note: In configuring the security for Guidewire Solr Extension batch load, follow the same instructions as for the corresponding Guidewire Solr Extension server type. In addition, configure a public certificate for the JVM running batch load in accordance with standard JVM properties. These property settings will be in either the `batchload.bat` file or the `batchload.sh` file.

Configuring Guidewire Solr Extension clients to run with SSL activated

In both standalone and cloud client configurations, configure security certificates and other environment variables. These settings include the `SOLR_HOME` variable. Configure the security certificates and environment variables in `solr.in.cmd` or `solr.in.sh` in accordance with Apache instructions.

See also

- *Configuration Guide*
- https://lucene.apache.org/solr/guide/6_6/enabling-ssl.html#enabling-ssl

Configuring 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. It is only possible to set up free-text search for embedded operation in development environments.

Set up free-text search for embedded operation on QuickStart

About this task

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

Procedure

1. In Studio **Project** window, expand **configuration→config→solr**:
 - a. Open `solrserver-config.xml`.
 - b. 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>
```

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

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

2. Save your changes to the `solrserver-config.xml` file.
3. Open a command prompt in the PolicyCenter installation directory and run the following command:


```
gwb packageSolr
```

Next steps

After completing this procedure, proceed to the *Configuration Guide*.

Set up free-text search for embedded operation on Tomcat

About this task

It is possible to set up free-text search in a development environment on Tomcat for either 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.

Procedure

1. In the Studio **Project** window, expand **configuration→config→solr**:
 - a. Open `solrserver-config.xml`.
 - b. 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>
```

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

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

2. Save your changes to the `solrserver-config.xml` file.
3. Open a command prompt in the PolicyCenter installation directory and run the following command:


```
gwb packageSolr
```

Next steps

After completing this procedure, proceed to the *Configuration Guide*.

Configure free-text search for external operation

Before you begin

PolicyCenter free-text search requires that you set up a Guidewire Solr Extension instance separate from the server instance that runs your PolicyCenter application.

Note:

The PolicyCenter base configuration presets a number of items of which two are of emphasis here. First, PolicyCenter automatically sets the SOLR_HOME environment variable with a relative path. In a Microsoft Windows context, the application configures the variable in the `solr.in.cmd` path. In a Linux or Mac context, the application configures the environment variable in the `solr.in.sh` path. You do not need to set the SOLR_HOME environment variable.

Second, the PolicyCenter base configuration accounts for logging. You do not need to change the logging settings to configure free-text search.

Procedure

1. On the host where the Guidewire Solr Extension instance for free-text search will reside, create an installation directory to use as the Guidewire Solr home directory. For example:

Unix – `/opt/gwsolr/pc`

Windows – `C:\opt\gwsolr\pc`

This topic uses `/opt/gwsolr/pc` as the Guidewire Solr home directory name.

2. Extract file `/solr/pc-gwsolr.zip` on the PolicyCenter host into the folder you created on the Guidewire Solr Extension host.
3. Start Guidewire Solr Extension by performing one of the following from the `\opt\gwsolr\pc\bin` directory:
 - If you want Guidewire Solr Extension to operate in standard fashion, run the following command:
`solr start -p`. In the `solr` command, the `-p` option indicates the port number.
 - If you want to access the Guidewire Solr Extension with a remote debugger, run the following command:

`solr start -p 8983 -a [Debug Arguments]`

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

4. Examine the logs to ensure the Guidewire Solr Extension web application started successfully. Find the logs at `\opt\gwsolr\pc\server\logs`. One such log file is `solr.log`.
The application started successfully if you see a number of log entries related to `pc-policy-active`.
5. In a browser, open the administrative user interface for the Guidewire Solr Extension web application by entering the following URL:

```
http://hostName:8983/solr
```

6. Verify that you see links to administrative pages for each entity type that is searchable in PolicyCenter with free-text search.

For example, click **Core Admin** to see information for the `pc_policy_active` core.

Next steps

After completing this procedure, proceed to “About the free-text batch load command” on page 119.

About the free-text batch load command

On Windows, ensure that 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 **Free-text Search** page from the **Internal Tools** tab.

Free-text search configuration files

Guidewire stores the configuration files that you modify during setup, including the batch load command itself, in the following directory:

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

If you created additional index document types, you need to modify the configuration files and the batch load command, in the following directory for each document type:

`/opt/gwsolr/pc/solr/documenttype_active/conf`

Guidewire locates most of the setup information for the free-text batch load command in a configuration file for your database brand. The configuration file name has the following format:

```
batchload-config-databaseBrand.xml
```

Free-text search configuration parameters

The following list describes the configuration parameters that you may need to modify in the `batchload-config` file:

Configuration parameter	Description
<code>absolutePathToPostprocessorExe</code>	Locates the <code>postprocess</code> 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 <code>/opt/gwsolr</code> .
<code>absolutePathToSortExe</code>	Location of a sort binary. You must modify this parameter if you use a binary other than <code>/bin/sort</code> on Unix or <code>c:\cygwin\bin\sort.exe</code> on Windows.
<code>absolutePathToSortTmpDir</code>	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 <code>absolutePathToWorkDir</code> .
<code>absolutePathToWorkDir</code>	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. Ensure the directory that you specify has sufficient high-performance disk space. If you change this value, you must make a matching change to <code>data-config.xml</code> .
<code>dataSource</code>	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.
<code>documentRoot</code>	Highest directory where documents for Guidewire Solr Extension are stored. The default value for this parameter is <code>\${solrRoot}/pc/solr/policy_active</code> .
<code>solrRoot</code>	Highest directory where Guidewire Solr Extension files are stored. The directory is where you extract the Guidewire Solr Extension .zip installation file. The default value for this parameter is <code>/opt/gwsolr</code> .
<code>solrServerConfigFile</code>	Locates the configuration file that controls how the free-text feature in PolicyCenter operates and interacts with the Guidewire Solr Extension. You must modify this parameter if your Guidewire Solr Extension home directory is other than <code>/opt/gwsolr</code> .
<code>xsi:noNamespaceSchemaLocation</code>	Location of the XSD for batch load configuration files. You must modify this parameter if your Guidewire Solr home directory is other than <code>/opt/gwsolr</code> .

For Windows paths, you can use the forward slash (/) character as the path separator. If your Guidewire Solr home directory is on a different Windows drive than your PolicyCenter installation directory, you must specify the drive letter as well.

Note: You can substitute the parameters in the preceding table from a property file or an environment variable.

See also

- *System Administration Guide*
- *Configuration Guide*

Configure the free-text batch load command for PolicyCenter

Procedure

1. Navigate to the following Guidewire Solr configuration directory:
`/opt/gwsolr/pc/solr/policy_active/conf`
2. Open the batchload shell script or batch file for editing and perform the following steps:
 - a. If using a directory for the Guidewire Solr home directory other than `/opt/gwsolr`, modify the `BASE_DIR` environment variable to locate the configuration files.

For example:

Unix

```
BASE_DIR=/opt/gwsolr
```

Windows

```
set BASE_DIR=C:\opt\gwsolr
```

- b. 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:

Unix

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

Windows

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

3. Open the batch load configuration file for which you specified a location in the previous step and perform the following steps:
 - a. Review the following elements for possible changes.

Unix

```
<param name="solrRoot" value="/opt/gwsolr"/>
<param name="absolutePathToSortExe" value="/bin/sort"/>
```

Windows

```
<param name="solrRoot" value="c:\opt\gwsolr"/>
<param name="absolutePathToSortExe" value="c:\cygwin\bin\sort.exe"/>
```

- b. 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"/>
```

IMPORTANT Due to security concerns, Guidewire recommends that you configure Guidewire Solr Extension database credentials to come from a property file or from environment variables. Do not hard code the database credentials in step 3b. Also due to security concerns, Guidewire recommends that if you use a property file for the database credentials, you do not place that file in either `C:\opt\gwsolr\pc\solr\policy_active\conf` or `C:\opt\gwsolr\pc\solr\policy_active`. Instead, place the property file in a folder adjacent to `C:\opt\gwsolr\pc\solr\policy_active\conf`.

4. On Windows, edit `postprocess.bat` and, if necessary, modify the `CoreUtils` environment variable to locate the sort binary.

For example:

```
set CoreUtils C:\cygwin
```

5. Restart the Solr application to pick up the changes.
6. Run the `batchload` command to test the setup.
7. 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.

Next steps

After completing this procedure, proceed to the *Configuration Guide*.

Command reference

PolicyCenter includes a number of command prompt tools that assist with build and administrative tasks on your PolicyCenter server. Because of the simplicity and power of the commands, command prompt tools are the preferred method of controlling server behavior, loading data, and generating tools in the PolicyCenter configuration environment.

The `gwb` command prompt tools are based on a build automation system called gradle. The `gwb` command prompt tool is a batch file. There are multiples gradle files that control how the `gwb` command prompt tools work. Do not modify these gradle files, because such modifications can break the build process.

See also

- *System Administration Guide*

Core application tasks

Command	Action
<code>gwb gwTasks</code>	Displays all Guidewire <code>gwb</code> command options.
<code>gwb clean</code>	Deletes the build directories.
<code>gwb cleanIdea</code>	Deletes the PolicyCenter Studio project files (files with <code>.iml</code> and <code>.idea</code> extensions).
<code>gwb codegen</code>	Generates metadata classes, page configuration classes, permission classes, localization classes, xml classes, and entity role constraints classes.
<code>gwb compile</code>	Compiles and copies resources for the QuickStart application server. Run this command before you start the QuickStart server. See the <code>runServer</code> command for more information.
<code>gwb dropDB -Env=env</code>	Prepares a new database for use by the Guidewire application. It will act upon the database configured in <code>database-config.xml</code>, or as optionally specified by the <code>-Env=env</code> parameter. Use caution whenever using this command. 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. You can pass additional parameters to this command by adding the parameters to the <code><reset-tools-params></code> subelement in <code>database-config.xml</code> as shown in the following code example:

```
<database>
  <dbcp-connection-pool>
    <reset-tool-params system-username="user" system-password="pass"/>
  </dbcp-connection-pool>
</database>
```

Command	Action
	<pre></dbcp-connection-pool> </database></pre> SQL Server The command drops and creates a new database. It creates any filegroups named within the <database> element. The database files will be physically located where the server would put them by default. If specified, the collate attribute specifies the database collation. Otherwise, the default collation of the database server will be used. The command SET READ_COMMITTED_SNAPSHOT ON WITH ROLLBACK IMMEDIATE is issued on the new database. Since a CREATE DATABASE command is used, attributes of the model system database of the server are inherited. The dropDB command will create any file groups listed in the upgrade element of the database configuration. This does not happen on Oracle. However, the dropDB 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 dropDB to create the database with file groups. 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 dropDB command requires SQLPlus, available with the Oracle client installation. Include ORACLE_HOME\bin in the PATH, so that dropDB can locate SQLPlus. The system-username and system-password attributes of the <reset-tools-params> element must be specified so that the required permissions are available for these actions.
gwb idea	Builds the PolicyCenter Studio project.
gwb inspect	Runs Guidewire Studio inspections.
[--include <i>include-dirs</i>]	include <i>include-dirs</i>
[--outputDir <i>output-dir</i>]	Comma separated list of paths to additional directories to include in the inspection analysis. The directories listed here must be located within modules/configuration, and the paths specified must be relative to modules/configuration. The following directories are always included by default:
[--profileName <i>profile-name</i>]	• gsrc • pcf/web • config/rules • config/resources/productmodel
[--profileXML <i>file-path</i>]	outputDir <i>output-dir</i>
[--scopeXML <i>file-path</i>]	The full path to an existing directory where the XML file with the inspection report will be stored.
[--verbose <i>level</i>]	profileName <i>profile-name</i>
	The name of the inspection profile to use for the inspection analysis. You can specify one of profileName or profileXML, but not both.
	profileXML <i>file-path</i>
	The path to the XML file containing the profile to use for the inspection analysis. You can specify one of profileName or profileXML, but not both.
	scopeXML <i>file-path</i>
	The path to the XML file containing the scope file to use for the inspection analysis.
	verbose <i>level</i>
	The logging level. Specify a number between 0 and 3, where 0 means no logging and 3 is frequent logging. The default is 0.

Command	Action
<code>gwb runServer -Denv=env</code>	Starts the bundled QuickStart application server. Depending on what you want to do, use the following command options: - To start the server in development mode using socket debugging, use the command <code>gwb runServer --debug-socket --no-suspend</code>. - To start the server in development mode in a suspended state using socket debugging, use the command <code>gwb runServer --debug-socket</code>. - To start the server in development mode using shared memory debugging, use the command <code>gwb runServer --debug-shmem</code>. - To start the server using a non-default port, use the command <code>gwb runServer -Dgw.port=nnnn</code>. - To start the server in development mode in a suspended state using shared memory debugging, use the command <code>gwb runServer --debug-shmem --no-suspend</code>. - To start the server in a particular environment, set the optional <code>-Denv="env"</code> parameter. IMPORTANT There is a dependency between the <code>gwb runServer</code> command and the <code>gwb compile</code> command. Guidewire recommends that you run the <code>gwb compile</code> command separately, before you run the <code>gwb runServer</code> command, for example: <pre>gwb compile gwb runServer -x compile</pre> Use the <code>-x compile</code> option with the <code>runServer</code> command to remove the dependency between the <code>compile</code> and the <code>runServer</code> commands. Otherwise, PolicyCenter must first verify what resources, if any, need to be recompiled, then perform an incremental recompile of those resources before starting the server.
<code>gwb stopServer</code>	Stops the bundled QuickStart application server.
<code>gwb studio</code>	Starts Guidewire Studio.

Configuration upgrade tasks

Command	Action
<code>gwb genRuleReport</code>	Builds the rule repository report.

Application server tasks

Command	Action
<code>gwb earWeblogicDbcp</code>	Builds the EAR file for WebLogic including JDBC drivers. Use <code>gwb earWeblogicDbcp</code> if you are going to have PolicyCenter manage the database connection pool.
<code>gwb earWeblogicJndi</code>	Builds the EAR file for WebLogic without JDBC drivers. Use <code>gwb earWeblogicJndi</code> only if you are going to use a JNDI database connection managed by WebLogic.
<code>gwb earWebSphereDbcp</code>	Builds the EAR file for WebSphere including JDBC drivers. Use <code>gwb earWebSphereDbcp</code> if you are going to have PolicyCenter manage the database connection pool.
<code>gwb earWebSphereJndi</code>	Builds the EAR file for WebSphere without JDBC drivers. Use <code>gwb earWebSphereJndi</code> only if you are going to use a JNDI database connection managed by WebSphere.
<code>gwb warJbossDbcp</code>	Builds the generic WAR file for JBoss including JDBC drivers. Use <code>gwb warJbossDbcp</code> if you are going to have PolicyCenter manage the database connection pool. You can include the Boolean parameter <code>includeDictionary=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: <pre>gwb warJbossDbcp -DincludeDictionary=true</pre>

Command	Action
	If includeDictionary=true, the command creates a dictionary folder within the WAR file. The dictionary folder contains the following subfolders: • data • security 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.
gwb warJbossJndi	Builds the generic WAR file for JBoss without JDBC drivers. Use gwb warJbossJndi only if you are going to use a JNDI database connection managed by JBoss. You can include the Boolean parameter includeDictionary=true, which also generates the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command: <pre>gwb warJbossJndi -DincludeDictionary=true</pre> If includeDictionary=true, the command creates a dictionary folder within the WAR file. The dictionary folder contains the following subfolders: • data • security 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.
gwb warTomcatDbcp	Builds the generic WAR file for Tomcat including JDBC drivers. Use gwb warTomcatDbcp if you are going to have PolicyCenter manage the database connection pool. You can include the Boolean parameter includeDictionary=true to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command: <pre>gwb warTomcatDbcp -DincludeDictionary=true</pre> If includeDictionary=true, the command creates a dictionary folder within the WAR file. The dictionary folder contains the following subfolders: • data • security 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.
gwb warTomcatJndi	Builds the generic WAR file for Tomcat without JDBC drivers. Use gwb warTomcatJndi only if you are going to use a JNDI database connection managed by JBoss. You can include the Boolean parameter includeDictionary=true to also generate the PolicyCenter <i>Data Dictionary</i> and <i>Security Dictionary</i> while building the WAR file. Use the following command: <pre>gwb warTomcatJndi -DincludeDictionary=true</pre> If includeDictionary=true, the command creates a dictionary folder within the WAR file. The dictionary folder contains the following subfolders: • data • security 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.

See also

- “Using a JNDI data source” on page 71

Globalization tasks

Command	Action
<code>gwb diffDisplayKeys</code>	A display key difference tool that does the following: - Compares each locale configured on the server against the master display key list. - Generates a file that contains a list of any missing keys.
<code>gwb exportLocalizations -Dexport.file=translation_file -Dexport.language=export_language</code>	Exports a translation file from PolicyCenter into a file. The <code>-Dexport.file</code> parameter specifies the destination file: - 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. The <code>-Dexport.language</code> parameter specifies the destination language to export. The <code>-Dexport.language</code> parameter must match a PolicyCenter <code>LanguageType</code> typecode, such as <code>fr</code> or <code>ja</code> .
<code>gwb importLocalizations -Dimport.file="translation_file" -Dimport.language=destination_language</code>	Imports a translation file into the configuration. 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. The <code>-Dimport.language</code> parameter specifies the destination language for the translations. The language must match a PolicyCenter <code>LanguageType</code> typecode, such as <code>fr</code> or <code>ja</code> .

See also

- Globalization Guide*

Integration tasks

Command	Action
<code>gwb genExternalSchemas</code>	Generates external JSON, Swagger SJON, and XSD schemas for internal integration JSON and Swagger schemas.
<code>gwb genFromWsc</code>	Builds WSC meta-information. Run this command whenever there are new <code>.wsc</code> files containing web service URLs available to generate the web service stub code. Places all WSC files in the configuration module.
<code>gwb genWsiLocal</code>	Generates the WSDL for WSI web services in <code>gsrc/wsi/local</code> .
<code>gwb exportWsd1</code>	Exports WSI web service WSDL files and related resources to a single JAR file. The generated JAR file is located in the directory <code>modules/configuration/build</code> . The individual WSDL files are generated in subdirectories of <code>modules/configuration/build/wsd1/wsi/wsd1</code> . For a client program to access the published services, the Java <code>wsimport</code> tool can parse the WSDL files and generate the necessary JAX-WS files. The operation can be automated by a Maven POM file.
<code>gwb jsonSchemaCodegen</code>	Generates code for the JSON schemas in the file, <code>codegen-schemas.txt</code> .
<code>gwb updateReleasedSchemaVersions</code>	Adds a versioned Integration View or REST API schema to <code>versions.json</code> .

See also

- Integration Guide*

Change verification tools

Command	Action
<code>gwb verifyExtConfig</code>	Performs a verification of external property substitution to find errors. See the <i>System Administration Guide</i> for more information.
<code>gwb verifyResources [-Dresource.types=<types>]</code>	Verifies Gosu types, GX models, PCF files, WS-I web service annotations, XML schemas, and XML workflow files. You can use the <code>-Dresource.types=<types></code> parameter to verify only some of the resource types. The resource type options are: <code>annotation,gxmodel,pcf,workflow</code>. The types are not case sensitive. If you specify multiple types, use commas. If you include spaces in your list of types, surround the typelist with double-quotes. Examples: <code>gwb verifyResources -Dresource.types="annotation, pcf"</code> <code>gwb verifyResources -Dresource.types=pcf</code>
<code>gwb zipChangedConfig -DoutputFile=filename.zip [-DappRootDirectory application_home][-Dexclude "directory1;directory2"]</code>	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 PolicyCenter installation directory as the root. The tool saves the output file relative to the application root. This file must not already exist. Specify any directories to exclude by setting the <code>-Dexclude</code> parameter, which takes a quoted, semicolon delimited, list of directories to exclude. Note that you can only exclude directories, and cannot exclude files.

Documentation generation and other tools

Command	Action
<code>gwb ccTypelistGen</code>	Runs the ClaimCenter Typelist Generator to help you synchronize your ClaimCenter line of business model with your PolicyCenter product model.
<code>gwb genDataDictionary -DoutputFormat={html xml}</code>	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. Guidewire generate the dictionaries the first time you unzip PolicyCenter. Regenerate the dictionaries each time you update the data model. Run the <code>gwb genJavaApi</code> command before regenerating the security and data dictionaries. The <code>outputFormat</code> parameter is optional. If you omit this parameter, the default output format is HTML.

Command	Action
	You can also generate the dictionaries in HTML format while building a WAR file. See the descriptions for WAR tools in this command list for instructions. This command has the following useful options: - Set -DoutputFormat to html to generate HTML: PolicyCenter/dictionary/data/index.html PolicyCenter/dictionary/security/index.html - Set -DoutputFormat to xml to generate XML: PolicyCenter/build/dictionary/data/entityModel.xml PolicyCenter/build/dictionary/data/entityModel.xsd PolicyCenter/build/dictionary/security/securityDictionary.xml PolicyCenter/build/dictionary/security/securityDictionary.xsd
<code>gwb genDataMapping -Dsplit={true false}</code>	Builds the data mapping files. Data mapping files represent fields present in the physical database. None of the virtual fields are represented. Set -Dsplit=true to build the files split out by table and typelist. Set -Dsplit=false to build the data mapping files with all tables and typelists concatenated.
<code>gwb genEntityModelXml</code>	Generates the Data Dictionary in XML format with an associated DTD so the XML document can be translated.
<code>gwb genImportAdminDataXsd</code>	Regenerates the XSD files for importing administrative data.
<code>gwb genPcfMapping</code>	Builds the PCF mappings.
<code>gwb genPhoneMetadata</code>	Regenerates phone metadata in directory config/phone/data. Run this command if you have modified the phone metadata XML files
<code>gwb genPhoneMetadata</code>	Regenerates phone metadata in directory config/phone/data. Run this command if you have modified the phone metadata XML files
<code>gwb genRuleReport</code>	Generates an XML report describing the existing Gosu rules.
<code>gwb packageSolr</code>	Regenerates the Solr installation Zip file, pc-gwsolr.zip for deployment of the Guidewire Solr Extension.
<code>gwb runSuite -Dsuite=TestSuiteClassName</code>	Runs the tests in a specified test suite.
<code>gwb verifyExtConfig</code>	Generates a report that includes the number of server configuration files affected, the number of substitutions involved, and the status of plugin ExternalConfigurationProviderPlugin if disabled.
<code>gwb version</code>	Prints the product version.

See also

- *Configuration Guide*
- *Rules Guide*
- *Gosu Reference Guide*

Plugin development tasks

Command	Action
<code>gwb genJavaApi</code>	Builds the Java API libraries to the PolicyCenter/java-api directory.
<code>gwb pluginStudio</code>	Starts IntelliJ IDEA with OSGi Editor. This is a specially-configured instance of IntelliJ IDEA and included with Guidewire PolicyCenter. You can use this IDE for Java plugin development.

See also

- *Integration Guide*

User interface tasks

Command	Action
<code>gwb webResources</code>	Generates expanded, CSS files that you can debug from the Sass files that define style rules for PolicyCenter.

See also

- *Configuration Guide*

JVM options for gwb build commands

The following JVM options work with many of the PolicyCenter `gwb` build commands.

JVM option syntax	Description
<code>-Denv=aaaa</code>	Starts the PolicyCenter server using the specified environment variable (env).
<code>-Dgw.passthrough.systemProperty=dddd</code>	Starts the PolicyCenter server using the provided system property value. The option parameters have the following meanings: • <code>systemProperty</code> – The name of a system property defined in the <code><registry></code> element in file <code>config.xml</code>. • <code>dddd</code> – The value of the system property.

JVM env string value always lower case

If you use a JVM option to set the server environment, PolicyCenter converts the provided string to lower case automatically. For example, the conversion to lower case occurs if you use the following JVM options:

- Jetty - `-Denv`
- Tomcat - `-Dgw.pc.env`
- Studio - `-Dgw.pc.serverid`

Thus, if you enter `-Dgw.pc.env=TEST`, PolicyCenter converts the string `TEST` to the string `test`. Be aware that the automatic conversion of the env string to lower case can cause a referenced env value to not match the defined environment name.

Note: If you start the Quickstart server from the command line, use `-Doption`. If you use any other application server, including the server that starts from within Guidewire Studio, use `-Dgw.pc.option`

Reference table of core gwb commands

It is possible to use the `-Denv` and the `-Dgw.passthrough` JVM options with some, but not all, of the `gwb` build commands. The following table indicates whether the listed JVM command options work with the core `gwb` commands (tasks).

Core task	Can use JVM option	Cannot use JVM option
<code>clean</code>		•
<code>cleanIdea</code>		•
<code>codegen</code>		•

Core task	Can use JVM option	Cannot use JVM option
compile		•
dropDb	•	
genDataDictionary	•	
idea		•
runServer	•	
stopServer	•	
studio		•

JVM option examples

For example, to pass `-DmySystemProperty=someValue` to build command `dropDB`, use the following command option.

```
gwb dropDb -Dgw.passthrough.mySystemProperty=someValue
```

Then, to make the `dropDB` command specific to a test environment, use the following command (for a Jetty server).

```
gwb dropDb -Denv=test -Dgw.passthrough.mySystemProperty=someValue
```

The following build command illustrates the use of a pass through system property.

```
gwb runServer -Dgw.passthrough.javax.net.ssl.keyStore=/Java/jdk1.8.0_74/jre/lib/security/cacerts
```

