PointBase

System Guide

Version 4.8

Proprietary and Trademark Information

Table of Contents

Preface **5**
Purpose 5
Audience 5
Release Notes 5
Document Feedback 6
Document Conventions Used in This Guide 6

Before You Begin **7**
Java Virtual Machine Requirement 7
PointBase Jar Files 7
Understanding CLASSPATH 8
Understanding How CLASSPATH Works 8
Setting the CLASSPATH 9
Using JAR and ZIP files 10

Starting PointBase **11**
PointBase Commander and Console 11
Starting PointBase Commander or Console for the First Time 12

Advanced Tips for Starting PointBase **13**
Tips for Starting PointBase Server using Embedded - Server Option 13
Tips for Starting PointBase Embedded or Client 14
Tips for Security Manager 20

PointBase Supporting Tools **21**
PointBase Commander 21
PointBase Console 23
PointBase Index Consistency Checking Utility 23

Verifying PointBase **26**
PointBase Commander 26
PointBase Console 26
A JDBC Application 27
PointBase Example 27

Configuring PointBase **28**

PointBase.ini File 28
Configuring Database Properties 30

Performance Tuning 37
Optimizing Query Expressions 37
Optimizing MIN and MAX Functions in a Query 41
Optimizing Count(*) in a Query 42
Tuning Database Properties 43
Minimizing Locking Times 46
Using Indexes 46
Selecting Multiple Rows With Different Key Values 47
Using the UNION Operator 48

PointBase™ Cryptography 49
Database Encryption 49
Client and Server Communication Encryption 49
Available Algorithms 50
Setting Database Encryption 50
Setting Client and Server Communication Encryption 51

Database Log Flushing 52
How to Flush the Log Setting the log.filesize 52
How to Flush the Log Using PointBase Commander 53

PointBase™ Internationalization 54
What Types of Businesses are Likely to Use Unicode? 55
Are There any Limitations to Unicode? 55
Implementation 55
Restrictions 57

Using PointBase™ With IDEs 58
SunTM ONE Studio 58
Borland JBuilder 59
IBM VisualAge 60
WebGain Visual Cafe 4.1 60

Appendix A: System Tables 61

Appendix B: Error Messages 74

Appendix C:
Country and Language Codes 90
Language Codes 90
Country Codes 95

Preface

Thank you for your interest in Version 4.8 of the PointBase product line.

Purpose

This guide describes how to start and configure your PointBase RDBMS. The following is a list of some things you can expect from this guide.

- How to begin using PointBase Embedded and Embedded - Server Option
- A description of PointBase Embedded properties and how to configure
- PointBase Cryptography

Audience

This guide is geared towards the Java developing community. Because PointBase is a 100% Pure Java Application Database, this guide assumes that you have the following concepts:

- basic knowledge of the Standard Query Language (SQL).
- basic knowledge of the Java programming language.
- basic knowledge of Java Database Connectivity (JDBC).
- basic database concepts.

Release Notes

The following link displays the most up-to-date information on PointBase products.

www.pointbase.com/support/releasenotes.html

Document Feedback

Please send comments or suggestions for all PointBase documentation to the following email address.

pbdocfeedback@pointbase.com

Document Conventions Used in This Guide

Convention	***Identifies***	***Examples***
ALL UPPERCASE LETTERS	• Environment variables • Database table names • SQL Keywords	• PATH • S_LST_OF_VAL • CREATE TABLE
`Courier New font`	• Directory, file, folder, and path names • Code • Data you need to type	• `c:\pointbase\img.bmp` • `Set PointBase =` • `Type Your Company Name Here`
Initial Uppercase Letters	PointBase names, objects, properties, windows, screens, dialog boxes, menus, buttons, tabs, applets, fields, and icons	PointBase Embedded, Business Component object, List Editor window, Main menu, and Cancel button
Italics	• Book titles • Cross references in an index or glossary • Variables • Arguments to statements of functions • First appearance of a new word or phrase • Emphasis	• *User's Guide* • *see also* or *see* • *APPSRVR_4X_ROOT* • *variable, rate, prompt$* • *new word* or *phrase* • Do not do this *before* you do that.
[ ]	Optional italicized arguments or characters inside angle brackets	[*caption$*]
{ | }	Choice from listed arguments; use OR operator (|) to separate	{Goto *label* | Resume Next | Goto 0}

Before You Begin

This section describes important information about the Java Virtual Machine, Jar files and CLASSPATH usage. If you are familiar with Jar files and CLASSPATH usage you can continue to the next chapter. However, it is important to understand the concepts covered in this chapter before you start PointBase.

Java Virtual Machine Requirement

PointBase Embedded and PointBase Embedded - Server Option require the use of a Java Virtual Machine (JVM) compliant with the JDK 1.2 specification or higher.

PointBase recommends using the latest JVM available in order to be able to use all of the functionality provided by PointBase. For example, if you were to use a JDK 1.2-compliant JVM, PointBase would work, but some JDBC 3.0 methods would not be available.

PointBase Jar Files

PointBase appends a version number to the end of every PointBase jar file. A version number is two digits. PointBase Embedded contains the `pbembedded48ev.jar` file. PointBase Embedded is a database that accepts multiple concurrent connections from a “single” client application. It is designed to run in the “same” JVM as the application of the developer. The small footprint of PointBase Embedded makes it ideal for embedding within applications that require built-in database functionality.

PointBase Embedded - Server Option contains both the `pbclient48ev.jar` file and the `pbembedded48ev.jar` file. PointBase Embedded - Server Option is a database server that accepts multiple concurrent connections through a network from client applications. It is designed to run in a standalone JVM with the client applications running in separate JVMs, which use the `pbclient48ev.jar`. The client applications can reside on the same machine as the server or on different machines on the network.

For more information about the architecture of PointBase Embedded, please refer to the *PointBase Developer’s Guide*.

Understanding CLASSPATH

The following sections describe CLASSPATH concepts in PointBase which are essential to using the PointBase jar files. CLASSPATH is either an environment variable or a Java Virtual Machine (JVM) command line option, such as -classpath on Sun JVM. The CLASSPATH tells the JVM where to find the files that it requires on the file system and is similar to the DOS Path variable.

Understanding How CLASSPATH Works

A Java compiler (e.g. javac or jvc) compiles Java source files into class files, which a JVM is able to interpret and execute. The JVM loads classes based on their name and the location indicated by the CLASSPATH provided. For example, suppose our CLASSPATH is

```
classpath=c:\qa;c:\production
```

In this example, the system looks at only two directories: “qa” first, and then “production” if the required class files are not found.

When the system becomes too large, developers create packages to group together related code. These packages become part of the class’ name. For example, for a given file databaseSample.java, the package statement may be

```
package com.pointbase.samples;
```

Where the fully qualified class name of the class file is the following:

```
com.pointbase.samples.databaseSample
```

The class loader appends “.class” to the class name and replaces the dots with system-specific path delimiters. If you type the following:

```
java com.pointbase.samples.databaseSample
```

The JVM adds path delimiters such as the following:

```
com\pointbase\sample\databaseSample.class
```

The system runs through the CLASSPATH from left to right in attempting to locate the files. Using the classpath example above, it first searches for:

```
c:\qa\com\producer\fruits\Banana.class
```

If successful, the system uses the class files found. If it fails to locate the file, it moves on to the next folder specified in the CLASSPATH. In our example this would be the following:

```
c:\production\com\producer\fruits\Banana.class
```

If the class loader cannot find the required class file in any of the directories specified in the classpath, the JVM reports that the class could not be found.

Setting the CLASSPATH

You may set the classpath as an operating system environment variable or as a JVM command line option. The following sections describe how to set the CLASSPATH, both permanently and temporarily, for a number of common operating systems.

Using a Java Virtual Machine running Windows NT or Windows 2000

- Select the Environment tab via the Control Panel. (On windows 2000 you access the Environment Variables through the Advanced tab.)
- Select or create the CLASSPATH variable and modify its value appropriately. Semi-colons delimit class path values.
- Click Set (or OK on windows 2000), then Apply to complete the entry.

To temporarily create or update your CLASSPATH environment variable, open a command prompt window and type:

```
set classpath=path1;path2;
```

The following is an example:

```
set classpath=c:\qa;c:\production;
```

Using a Java Virtual Machine running Windows 95/98

You need to update your autoexec.bat file with the path of the JVM libraries in the CLASSPATH environment variable.

To temporarily create or update your CLASSPATH environment variable, open a command prompt window and type:

```
set classpath=path1;path2;
```

The following is an example:

```
set classpath=c:\qa;c:\production;
```

Using UNIX Java Virtual Machine running Linux, Solaris, etc.

You need to update your `.login` or `.profile` file with the path of the JVM libraries in the CLASSPATH environment variable.

To temporarily create or update your CLASSPATH environment variable, open an operating system window and type:

```
setenv classpath /path1:/path2
```

The following is an example:

```
setenv classpath /qa:/production
```

Using JAR and ZIP files

Java Virtual Machines allow you to group together classes into one file, called either a jar file or a zip file. Both file formats may be compressed. In addition, jar files may also contain a certificate that digitally signs the originating location of the jar file.

To use a jar file, you must specify its location to the JVM using the CLASSPATH. Continuing with the example above, if you package the databaseSample class into a file called Pointbase.jar, the previously defined CLASSPATH will not help in finding the class because the JVM does not know about the jar file. You must specify jar files explicitly in the classpath, as in the following example:

```
classpath=c:\Pointbase.jar
```

If you have specified the following class:

```
java com.pointbase.samples.databaseSample
```

then the JVM now looks at the CLASSPATH and finds the specified jar file. When found, the jar file is opened by the JVM and inspected to verify there is a class inside called:

```
com.pointbase.samples.databaseSample.
```

The class is then directly loaded from the jar file.

Starting PointBase

The following sections describe how to start PointBase. You can start PointBase in two different ways:

- Using PointBase Commander
- Using PointBase Console

PointBase Commander and Console

PointBase Commander is a command line tool that you can use to perform administration tasks, such as creating tables and running SQL scripts. As a database client, PointBase Commander is oriented towards batch mode operation. Note that every command ends with a semicolon in PointBase Commander.

NOTE: Currently, PointBase Commander has not been fully tested on a Macintosh.

PointBase Console is a graphical user interface (GUI) tool. With PointBase Console, you can administer your data and perform the following:

- Create new databases.
- Execute, debug, and display the results of SQL commands.
- Create text filters from databases.
- Show the results of SELECT statements graphically.
- Import and export data.

NOTE: This application uses Sun's JFC/Swing package which is automatically downloaded when you install PointBase.

Starting PointBase Commander or Console for the First Time

You can start the Commander or Console tool for PointBase Embedded or the Client by completing the following steps. However, to launch the Commander or Console for the Client, you must first start the PointBase Server to use PointBase Embedded - Server Option.

Using Microsoft Windows

- Click the Start button on the Task bar.
- Navigate to programs, where you will find and select PointBase 4.7.
- Select Tools.
- Select Embedded Commander or Console *or*
- Select Start Server, and then select Server Commander or Console.

Using a Command Line Window

You can start the Commander or Console tool for PointBase Embedded or Server Option by using a command line window and executing one of the PointBase executable files, for example, “embedded_commander.exe” or “embedded_console.exe” for the Windows OS. However, you must execute the “start_server” executable file before you execute the “server_commander” or “server_console” files. By default, the location of the executable files are in the directory “<install_directory>\tools\<pointbase product>.” For example, you would navigate to the following file if you are using the Windows OS:

```
c:\pointbase\tools\embedded\embedded_commander.exe
```

Advanced Tips for Starting PointBase

The following sections describe how you can start PointBase immediately, without having to answer the previous questions every time you launch PointBase. You can use a command line to specify everything you need to start using PointBase, for example, the PointBase product you want to launch (e.g. PointBase Embedded or Embedded - Server Option), what database to access, and which tool (Commander or Console) you want to use. The commands may vary, depending on your JVM.

Tips for Starting PointBase Server using Embedded - Server Option

You can edit the `bat` file located in the "<install directory>\pointbase\samples\server_embedded\" directory before starting the PointBase Server. By editing the bat file you can add commands at the end of the script (after "com.pointbase.net.netServer"). To edit the bat file, open it in any text editor and append any of the following commands to the end of the script and execute the `bat` file, for example, enter the following:

```
java com.pointbase.net.netServer /win
```

- `/win` starts the server window. The system brings up a window displaying the server status. Use this window to collect statistical information on usage, to log information, or to shutdown the server.
- `/d:<level>` displays level (from 0 to 3). The server echoes JDBC API calls. Level 3 displays the most details.
- `/database:<name>` pre-opens the named database for an efficient first connection.
- `/port:<port>` specifies the port number to listen on.
- `/file:<filename>` records the netserver log to the specified filename.
- `/pointbase.ini=<path to pointbase.ini>` configures the system to search for the "pointbase.ini" file in the path specified.
- `/help` displays all available command lines.
- `/noconsole does not display the netServer window, allowing you to run PointBase Server as a background thread on UNIX and Solaris platforms.`

You can also enter the following commands in the Server command line window *after* you launch the `bat` file.

- `h or?` displays the possible commands for this window.
- `x` or `q` shuts down the server.
- `l` displays the locked tables.
- `c` shows currently active connections, and for each connection shows the active statements, and for each statement shows the actual SQL code.
- `v` displays the server version number.

Tips for Starting PointBase Embedded or Client

PointBase recommends that you create an executable file such as a DOS bat file (or UNIX shell script) to launch the product, so you do not have to retype the launch script each time you start the product. You can use, for example, the bat files in the directory: “<install directory>\pointbase\samples\server_embedded.”

In order to create an executable file to launch PointBase, you must know the location of the PointBase JAR files.

Locating Jar Files

By default the `jar` files are located in the following directory “\pointbase\lib.” For PointBase Embedded, the JAR’s are “pbembedded48.jar and pbtools48.jar,” and for PointBase Server they are “pbserver48.jar,” “pbclient48.jar,” and “pbtools48.jar.”

Calling Tool Class Files

To call a specific tool (Commander or Console) in your executable file, you must specify the class at the command line in addition to the `jar` files, url, driver, etc. You will see in the following examples, PointBase specifies the tool classes by the following entries: com.pointbase.tools.toolsCommander (for Commander) or com.pointbase.tools.toolsConsole (for Console). You may interchange these classes, depending on your preference.

NOTE: You must include the `swingall.jar` file, when you call the Console class, com.pointbase.tools.toolsConsole. For example:

```
java -classpath pbembeddedxx.jar;pbtoolsxx.jar
com.pointbase.tools.toolsConsole [-plugin] driver url [,parameters] username password
```

Variable Descriptions

The following table describes the variables in the examples to follow.

Variable	Description
database	Embedded, or Client (for example, pbembedded48)
xx	The release number of the product
com.pointbase.tools.toolsCommander	The tool class file for Commander
com.pointbase.tools.toolsConsole	The tool class file for Console
plugin	Used with special operating systems that are embedding the Java Virtual Machine. Specifying this option does not terminate the Java Virtual Machine when closing the Console window. Use this option for Console only.
url	Describes the location of the database to which you connect. Possible URLs are: • PointBase Embedded: jdbc:pointbase:embedded:*database_name* (PointBase also supports version 3.5 URLs or earlier.) • PointBase Embedded - Server Option jdbc:pointbase:server://machine_name<:port>/ *database_name* (PointBase also supports version 3.5 URLs or earlier.) *database_name* is the name of the database to which you connect. The default *port_number* is 9092. **NOTE:** If both server and client run on the same network machine, you may use LOCALHOST as the *database_name*.

Variable	Description
parameters	Flags to create a new database, to **overwrite** an existing database, or to connect to an existing database. The following are the supported flags: • *new*—creates a new database and connects to it, or connects to a database if it already exists. • *create=true*—behaves the same as the *new* flag. • *create=false*—connects to database if it already exists, or throws an exception if the database does not exist. • *create=force*—creates a new database and connects to it, or overwrites the database if it already exists—but only if you have authorization do to so. That is, you must be either the database owner or the PBSYSADMIN user. (See *PointBase Developer's Guide* for details on PBSYSADMIN predefined user.) If you specify more than one flag, PointBase will recognize the last flag you specify. In addition to, or instead of using one of the previous flags for your parameter, you can also define one or more database properties, for example: `database.home=c:\pointbase\myDatabases` The above example uses the following syntax: <database property> = <value> Refer to "Database Properties Described" on page 32 for more details about database properties.
script	The name of the script to run automatically. Use this option for Commander only.
user	The name of the user logging into the database (default: `PBPUBLIC`)
password	The password for the user (default: `PBPUBLIC`).
autocommit	The switch command to set autocommit on or off. Enter TRUE to set autocommit on, and enter FALSE to set autocommit off. If you do not define TRUE or FALSE, PointBase defaults to TRUE (autocommit on).

Using a Sun or IBM Java Virtual Machine

The following examples present a model for creating an executable batch or script file using a Sun or IBM Java Virtual Machine.

To start Commander

```
java -classpath pbembeddedxx.jar;pbtoolsxx.jar com.pointbase.tools.toolsCommander
com.pointbase.jdbc.jdbcUniversalDriver url[,parameters][script] user password
[autocommit]
```

To start Console

```
java -classpath pbembeddedxx.jar;pbtoolsxx.jar com.pointbase.tools.toolsConsole
com.pointbase.jdbc.jdbcUniversalDriver url[,parameters] user password [autocommit]
```

- To start Commander using PointBase Embedded (for example):

```
java -classpath .\lib\pbtools48.jar;.\lib\pbembedded48.jar
com.pointbase.tools.toolsCommander com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:embedded:sample,create=force pbpublic pbpublic true
```

To start PointBase Console using PointBase Embedded (for example):

```
java -classpath .\lib\pbembedded48.jar;.\lib\pbtools48.jar
com.pointbase.tools.toolsConsole com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:embedded:sample,create=false pbpublic pbpublic true
```

- To start Commander using the Client (for example):

```
java -classpath .\lib\pbtools48.jar;.\lib\pbclient48.jar
com.pointbase.tools.toolsCommander com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:server://server1/sample,database.home=c:\pointbase\myDatabases pbpublic
pbpublic false
```

To start PointBase Console using the Client (for example):

```
java -classpath .\lib\pbclient48.jar;.\lib\pbtools48.jar
com.pointbase.tools.toolsConsole com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:server://server1:9092/
sample,create=false,database.home=c:\pointbase\myDatabases pbpublic pbpublic false
```

Using a UNIX Java Virtual Machine running Linux, Solaris, etc.

The following examples present a model of how to create an executable file using a UNIX Java Virtual Machine running Linux, Solaris, or other flavors of UNIX.

To start Commander

```
/bin/java -classpath pbtoolsxx.jar:pbembeddedxx.jar com.pointbase.tools.toolsCommander
com.pointbase.jdbc.jdbcUniversalDriver url[,parameters][ script] username password
[autocommit]
```

To start Console

```
java -classpath ./lib/pbembeddedxx:./lib/pbtoolsxx.jar com.pointbase.tools.toolsConsole
[-plugin] com.pointbase.jdbc.jdbcUniversalDriver url[,parameters] username password
[autocommit]
```

- To start PointBase Commander using PointBase Embedded (for example):

```
/bin/java -classpath ./lib/pbembedded48.jar:./lib/pbtools48.jar
com.pointbase.tools.toolsCommander com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:embedded:sample,create=true pbpublic pbpublic true
```

To start PointBase Console using PointBase Embedded (for example):

```
/bin/java -classpath ./lib/pbembedded48.jar:./lib/pbtools48.jar
com.pointbase.tools.toolsConsole com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:embedded:sample,create=true pbpublic pbpublic true
```

- To start PointBase Commander using the Client (for example):

```
/bin/java -classpath ./lib/pbclient48.jar:./lib/pbtools48.jar
com.pointbase.tools.toolsCommander com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:server://server1:9092/sample,create=force pbpublic pbpublic false
```

To start PointBase Console using the Client (for example):

```
java -classpath ./lib/pbclient48.jar:./lib/pbtools48.jar
com.pointbase.tools.toolsConsole com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:server://server1/sample,create=force pbpublic pbpublic false
```

Using a Macintosh OS 9

To run PointBase Console on a Macintosh running OS X or later, use the steps as given for Unix environments.

To run PointBase Console on a Macintosh running OS 9 or earlier and connect to a PointBase database, use JBindery. Use the following steps to run Console:

Setting the class file to run

In the JBindery Command window, enter the name of the PointBase Console class file to run. The class file is: `com.pointbase.tools.toolsConsole`.

Setting the command parameters

Next, you need to set some command line arguments in the Optional parameters field. The complete set of arguments to pass is:

```
[<-plugin>] <driver> <url>[,<any properties, comma separated>]<userid> <password>
[autocommit]
```

For example:

```
com.pointbase.jdbc.jdbcUniversalDriver jdbc:pointbase:mydb,database.home=MyHD/pointbase/
databases pbpublic pbpublic
```

NOTE: The arguments are separated by spaces. If you have a space in your drive name or in a folder name, you should delimit the full URL with double quotes.

PointBase by default looks in `\pointbase\databases` for a database, which the Mac interprets as <hard drive>/<folder>. The define option used in the Sun JVM has no equivalent in the MRJ. You should pass the database.home property as part of the URL.

Console also has a login dialog that appears if these parameters are not passed on startup. You can specify a URL there.

Setting the CLASSPATH

Next, you must set a CLASSPATH to the PointBase `jar` files located in the folder /`pointbase/lib`. In JBindery Classpath window, push the Add `zip` File button, then navigate to the location of the PointBase `jar` files (`\pointbase\lib`) and select the `jar` files you need. Here are the PointBase `jar` files:

- `pbtoolsxx.jar`: contains the class files for the PointBase Console query tool
- `pbembeddedxx.jar`: contains the class files for an application to connect to PointBase Embedded, and for the multi-user database network server
- `pbclientxx.jar`: contains the class files for an application to connect to PointBase server if using the Server option.

You must also set a CLASSPATH to the copy of `swingall.jar` in the `MRJ` folder. You should not use the `swingall.jar` included with PointBase, if using Mac OS 9.x.

Running PointBase Embedded and Client

PointBase Embedded

To run PointBase Console against the PointBase Embedded, you need the file `pbembeddedxx.jar`, as well as the PointBase Tools `pbtoolsxx.jar`.

PointBase Embedded - Server Option

To run PointBase Console using the PointBase Client tools, you need the files `pbclientxx.jar` and `pbtoolsxx.jar` (where *xx* represents the release number. Check the actual name of the `jar` file to be sure you have the correct name.)

To run your own application that connects to a PointBase database using the Server Option, you only need the file `pbclientxx.jar`.

Running PointBase Server with Embedded - Server Option

The class file to run in order to start PointBase Server is `com.pointbase.net.netServer`. The CLASSPATH must include `pbembeddedxx.jar`. The command line can accept the argument "/win" in order to have the PointBase Server window appear. If not used, the server will run "headless".

JBindery Tip

JBindery lets you save your configuration as a JBindery configuration, or as an application. If you save it as a JBindery configuration, you can open it with the same settings and run it. If you save it as an application, you can run it without seeing the JBindery interface.

Tips for Security Manager

PointBase also can be run with Java Security Manager. With Java Security Manager enabled, user may need to grant the following permission to PointBase jars.

java.io.FilePermission	Read, write, delete permissions on the PointBase database.home directory. For a client process where PointBase database.home directory is on a server machine and not available to the client, permissions on the current working directory, the directory specified by environment variable java.io.tmpdir or the root directory may be required.
java.net.SocketPermission	Listen, accept, connect, resolve permissions on server listening port. The default is localhost:9092, and on port localhost:20000 to 21000 for PointBase system locks. For clients, permissions on server listening port, for example serverhost:9092, are required.
java.util.PropertyPermission	Read permission on system properties java.version, java.vendor, java.version, user.name, os.name, os.version, os.arch, java.io.tmpdir, file.tmpdir, java.class.path, file.separator, and all PointBase properties listed in PointBase configuration section.
com.pointbase.sp.spPermission	on classes used by store procedures or functions

PointBase Supporting Tools

This section describes the PointBase tools, PointBase Commander, PointBase Console and integrity checking tools. You can use the PointBase tools to access and manipulate PointBase, as well as to check database consistency and fix database corruptions.

PointBase Commander

After you launch the Commander, it prompts you for the following settings. You can choose to accept the default values or enter new ones.

```
Do you wish to create a “New/Overwrite” Database? [default: N]:
```

(If you enter Y for yes, make sure to use a name that is different from your existing database, or the existing database(s) will be destroyed and overwritten with no warning by a new empty database.)

```
Select product to connect with: Embedded (E), or Server (S)? [default: E]:
Please enter the driver to use: [default: [com.pointbase.jdbc.jdbcUniversalDriver]:
Please enter the URL to use: [default: [jdbc:pointbase:embedded:sample]
```

(PointBase supports URLS from version 3.5 or earlier.)

```
User name: [default: PBPUBLIC]:
```

(The user name must exist in the SYSUSERS table in order to connect to the database.)

```
Password: [default: PBPUBLIC]:
```

In addition to standard SQL commands, PointBase Commander supports the following commands, which are specific to PointBase Commander only. When using PointBase Commander you must use only forward slashes (`/`), where slashes are required.

NOTE: Use the following command to view the most up-to-date commands that are specific to PointBase Commander: `HELP;`

Table 1: Commands for PointBase Commander Only

Command Syntax	***Description***
SET SPOOL ON/OFF <file_name>	Direct the output to a file ON/OFF. If the file is not specified than Commander.spl file is created in the current user dir.
SET STOP_ON_ERROR ON	Stop execution of SQL script if error occurs
SET STOP_ON_ERROR OFF	Do not stop execution of SQL script if error occurs
SET AUTOCOMMIT ON/OFF	Set the autocommit mode to TRUE or FALSE.
SET TIMING ON	Begin showing timing for commands.
SET TIMING OFF	End showing timing for commands.
SET SCREENHEIGHT <value>	Set the number of lines per page.
SET SCREENWIDTH <value>	Set the number of characters per line.
SET PAUSE ON	Turn ON pause after every command.
SET PAUSE OFF	Turn OFF pause after every command.
SET DATA ON/OFF	Turn ON/OFF the display of result set data.
SET NULLS <value>	Set NULL value display.
SHOW SYSTEM	Report Database Meta Information.
SWITCHLOGFILE	Switch to a new database log file.
DESCRIBE <table>	Report Table Information.
UNLOAD TABLE <table> <file>	Unload a table in SQL format. <Table> is the name of the table to unload. <File> is the .sql file that you want to receive the scripts.
UNLOAD DATABASE <file> [PRESERVE]	Unload a database in SQL format. <File> is the .sql file that you want to receive the scripts. [PRESERVE] preserves the ownership of schemas, grantors in GRANT statements, and create ROLE owners. **NOTE:** It does not preserve the DATABASE OWNER. Whoever creates the new database is the database owner.
SHOW MEMORY	Report memory used by the Java virtual machine.
RUN <file>	Run a SQL script from the .sql file.
RUNJDBCMETA <methodName>(args...)	Run a method in the JDBC database meta data.
--	Comment out a single line, outside any SQL statement.
/* ... */	Comment out multiple lines, inside or outside, any SQL statement.

NOTE: Please see "Optimizing Query Expressions" on page 37 for additional commands.

PointBase Console

Please refer to the *PointBase Console Guide* for information about using the PointBase Console tool.

PointBase Index Consistency Checking Utility

This utility is used to check index consistency. It checks index structural consistency and is able to dump detail index page information. To understand these page dumps requires some knowledge of Pointbase internal and btree structure. This utility also provides a fix option to recreate the index if it happened to be corrupted. This utility must be run alone on a quiescent database, otherwise unexpected result may occur.

Utility Syntax

```
java com.pointbase.util.utilCheckIndex dbname {-n [schemaName.][tableName.]indexName |
-p controlPageId} [-d] [-toString] [-level n] [-fix] [-userId userid] [-password password]
```

Parameters:

dbName:

Name of database.

-all

Check all indexes.

-n [schemaName.][tableName.]indexName:

Using index name to specify the index to be checked.

-p controlPageId:

Using control page id to specify the index to be checked. This page id can be obtained from sysindexes entries, column indexFirstPage.

-d:

Dumping detail index page contents. This will dump index key entries in both vertical and horizontal order, otherwise only vertical order will be shown.

-toString:

Working with -d option to dump index key entries as string instead of bytes.

-level n:

Checking level. Level1 checks index page links and key order consistency. Level 2 checks row pointers in index key entries and crosses checking with table entries in addition to level 1 checking.

-fix:

Fixing the index. This option will fix a corrupted index. It does check the index first, if the index checking passes, this option is no-op.

-userId userid

Specifying connection userid. Default is "pbpublic" if this parameter is not specified.

-password password:

Specifying connection password. Default is "pbpublic" if this parameter is not specified.

API Syntax

Connection must be established before calling these api's.

Imported class:

import com.Pointbase.util.utilCheckIndex;

APIs:

```
public static boolean check( int p_ControlPageId, int
p_Level, PrintWriter p_Out )
Parameters:
p_ControlPageId: index control page(or first page) id.
p_Level: checking level.
p_Out:  output stream.
Return: true for consistent, false for inconsistent.

public static boolean check( String p_IndexName, int
p_Level, PrintWriter p_Out )
Parameters:
p_IndexName: index name.
p_Level:  checking level.
```

```
p_Out:  output stream.
Return: true for consistent, false for inconsistent.


public static void fix(Connection p_Con, String p_IndexName)
Parameters:
p_IndexName: index name.
```

Verifying PointBase

You can verify PointBase in the following four ways:

- "PointBase Commander"
- "PointBase Console"
- "A JDBC Application"
- "PointBase Examples"

PointBase Commander

To verify PointBase using PointBase Commander, proceed as follows:

1. Start PointBase Commander.
2. Type the following SQL Statement:

```
> select tablename from systables;
```

The system returns a list of system tables and catalogs, such as SYSSCHEMATA, SYSUSERS, SYSTABLES. If the list is missing, please contact your PointBase Support representative at http://pointbase.custhelp.com.

PointBase Console

To verify PointBase using PointBase Console, perform the following:

1. Start PointBase Console.
2. Deselect "User Only Tables" from the Catalog menu.
3. Click the Catalog button from the Catalog menu.
4. Extend the Schemas icon to display the PointBase icon.
5. Extend the PointBase icon to display the Tables icon.
6. Extend the Tables icon.

A list of system tables and catalogs display, such as SYSSCHEMATA, SYSUSERS, SYSTABLES. If the list is missing, please contact your PointBase Support representative at http://pointbase.custhelp.com.

A JDBC Application

To verify PointBase using JDBC APIs, proceed as follows:

Create a sample program that executes a SELECT SQL statement to retrieve the system catalogs or you can refer to the JDBC Tutorial chapters of the *PointBase Developer's Guide* that uses the PointBase Sample Database Application. An example of the SELECT statement is as follows:

```
> select tablename from systables;
```

Once you have retrieved the system catalogs correctly, the system displays a list of system tables and catalogs, such as SYSSCHEMATA, SYSUSERS, SYSTABLES, etc...If the list is incomplete or missing, please contact your PointBase Support representative at http://pointbase.custhelp.com.

PointBase Example

You can verify PointBase with the example located in the "\pointbase\samples\embedded_server folder." It provides you with the following:

- Sample Database Application: this illustrates the basics of connecting, inserting some data into the database and selecting it for output. It also describes more advanced JDBC operations using PointBase, for example, batch operations and scrollable result sets.
- Example Batch files: these files give an example of batch files that start PointBase Commander or Console. By viewing and editing these files in a text editor you can see how batch files start PointBase.

Configuring PointBase

This chapter explains how to configure PointBase database properties. Configuring database properties allow you to control some aspects of PointBase behavior. The following sections describe how to configure your PointBase Embedded RDBMS.

PointBase.ini File

The "pointbase.ini" file contains the PointBase database properties. You can select the "pointbase.ini" parameters to configure the database properties. By configuring the database properties, you can increase the performance of your system.

Changing the Location of the PointBase.ini File

The "pointbase.ini" file has one default location: "<install directory>\pointbase\tools\<product name>." You can override the default location of the "pointbase.ini" file by using the *Define* option or by adding a parameter to the URL when connecting to PointBase.

Using the Define Option

You must use the Define option when you start PointBase. The following examples set the location of the "pointbase.ini" file to "c:\pointbase\config\pointbase.ini."

- Using the Microsoft Java Virtual Machine

```
jview /d:pointbase.ini=<path>
```

If you want the system to search for the "pointbase.ini" file in the "config" folder, enter the following:

```
jview /d:pointbase.ini=c:\pointbase\config\pointbase.ini /cp
c:\pointbase\lib\pbtools47.jar;c:\pointbase\lib\pbembedded47.jar
com.pointbase.tools.toolsCommander com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:embedded:sample pbpublic pbpublic
```

- When using the Sun Java Virtual Machine

```
java -Dpointbase.ini=<path>
```

If you want the system to search for the “pointbase.ini” file in the “config” folder, enter the following:

```
java -Dpointbase.ini=c:\pointbase\config\pointbase.ini -classpath
c:\pointbase\lib\pbtools47.jar;c:\pointbase\lib\pbembedded47.jar
com.pointbase.tools.toolsCommander com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:embedded:sample pbpublic pbpublic
```

Using the URL

Use the URL when starting PointBase. The following example sets the location of the “pointbase.ini” file by appending a parameter to the URL.

```
"jdbc:pointbase:embedded:databaseName,pointbase.ini=<path to pointbase.ini>"
```

If you want the system to search for the “pointbase.ini” file in the “config” folder, enter the following:

```
java -classpath
c:\pointbase\lib\pbtools47.jar;c:\pointbase\lib\pbembedded47.jar
com.pointbase.tools.toolsCommander
com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:embedded:sample,pointbase.ini=c:\config\pointbase.ini pbpublic pbpublic
```

NOTE: Because PointBase Server Option does not use a URL on the server side, you can change the location of the “pointbase.ini” file by entering the following at the command line, when you start PointBase Server.

```
java com.pointbase.net.netServer /pointbase.ini=<path to pointbase.ini>.
```

How PointBase Locates the PointBase.ini File

PointBase searches for the “pointbase.ini” file in the following areas:

- URL
- `-Dpointbase.ini=<folder\file>` (or for jview `/d:pointbase.ini=x`)
- In the directory where the application loading the driver is located

NOTE: In the case of PointBase Server Option, PointBase searches the command line because it has no URL option. See “Changing the Location of the PointBase.ini File.”

If PointBase is unable to find the “pointbase.ini” file, it uses the default database properties defined in the section, “Database Properties.”

Configuring Database Properties

This section describes how to configure the properties of the PointBase database. You can configure or set the database properties in the following ways:

- Edit the parameters in the “pointbase.ini” file located in the directory, “<install directory>\tools\<pointbase_product>.”
- Add a parameter(s) to the URL, when starting PointBase. The following example sets the database.home parameter to C:\myDatabase.

```
"jdbc:pointbase:embedded:databaseName,create=true,database.home=C:\myDatabases"
```

The following code describes a more complete example:

```
java -classpath
c:\pointbase\embedded\lib\pbtools47.jar;c:\pointbase\embedded\lib\pbembedded47.jar
com.pointbase.tools.toolsCommander
com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:embedded:databaseName,create=true,database.home=C:\myDatabases pbpublic
pbpublic
```

- Use the PointBase Console. Refer to the *PointBase Console Guide* for more information.
- Use the Define option when starting PointBase. The following example sets the database.home parameter to C:\myDatabase.

```
java -Ddatabase.home=C:\myDatabase ...
```

The following code describes a more complete example:

```
java -Ddatabase.home=C:\myDatabase -classpath
c:\pointbase\lib\pbtools47.jar;c:\pointbase\lib\pbembedded47.jar
com.pointbase.tools.toolsCommander com.pointbase.jdbc.jdbcUniversalDriver
jdbc:pointbase:embedded:databaseName,create=true pbpublic pbpublic
```

What Method of Configuration Takes Precedence?

Because database properties can be set in different ways, PointBase prioritizes which database properties to use first, to avoid any conflicts. For example, if you define the database.home parameter to c:\myDatabase using the URL and you also edit the “pointbase.ini” file to define the database.home parameter to c:\temp, PointBase uses the URL definition instead of the property set in the “pointbase.ini.” The database properties defined in the URL take first precedence, then the properties defined in the “pointbase.ini” and finally, the properties specified in the Define option. The following is the order of precedence.

1. URL parameter
2. -D (Define option)
3. `pointbase.ini`.

INI Parameter Organization

The INI parameters are organized into three categories.

Connection

Some INI parameters are connection specific. Parameters specified through the URL override the parameters specified by other means. If the same parameter is specified in multiple ways then the order of precedence stated above is applied.

Database

Similar conditions apply for database parameters as for connection parameters. However only the first connection is allowed to setup the initial parameters for that database. Any of these parameters specified in subsequent connections are ignored.

Process

The user can set up the process-based initial parameters only once, at the time of the first connection to any database in that process. Any of the parameters specified in subsequent connections are ignored.

These categories and their related INI parameters are shown in table below:

Connection	***Database***	***Process***
crypto.communicationAlgorithm	Crypto.databaseAlgorithm	connection.convertUserInfoToUppercase
crypto.communicationKey	crypto.databaseKey	cache.checkpointinterval
fetch.blocksize	database.pagesize	cache.size
stream.checkSize	debug.log	documentation.home
transaction.isolationLevel	debug.logSize	server.port
	debug.logLevel	database.home
	locale.country	
	locale.language	
	locks.maxCount	
	locks.timeout	
	log.filesize	
	log.syncatcommit	
	max.connections	
	sort.size	

	table.pageReserve	
	connpool.size	
	SQLCaching.size	
	cursor.holdAcrossCommit	

Note: When using Server option, the INI parameters that are set on the client (URL, -D , INI file) will override the parameters that are set on the server.

For example, consider a case where you start the server as shown below

```
java  -Ddatabase.home=e:\myfolder\personal com.pointbase.net.netServer
```

and then start the client application as follows:

```
java -Ddatabase.home=e:\publicfolder com.pointbase.tools.toolsConsole
```

As database.home is categorized as a process INI parameter, the first client will set the database.home as e:\publicfolder. All subsequent clients will connect to database located in e:\publicfolder.

To avoid this, it is recommended that a pointbase.ini file on the client side contain only parameters that relate to the client such as documentation.home and sort.size , and set server-side parameters in pointbase.ini on the server.

Database Properties Described

The following table describes the PointBase parameters in the “pointbase.ini” file, which you can use to configure the database properties. If you do not specify a value for a setting, the system uses the default value listed in the following table.

Table 1: PointBase Parameters

Key Name	***Default***	***Description***
cache.checkpointinterval	10	The checkpoint interval in seconds
cache.size	2063	Maximum number, in pages, held in cache. Best performance is found when this value is a prime number.

Table 1: PointBase Parameters

Key Name	***Default***	***Description***
connection.convertUserInfoToUppercase	true	Indicates the behavior of usernames and passwords that are specified in JDBC connection(i.e getConnection()) methods. If this parameter is set to TRUE then the usernames and passwords specified in the JDBC connection methods, will be converted to uppercase. If this parameter is set to false then the usernames and passwords will be taken as specified. The username and password for the default user PBPUBLIC will always be converted to uppercase regardless of the value of this parameter.
connpool.size	10	The number of connections kept in pool when using jdbcPooledDatasource to get connections.
crypto.communicationAlgorithm	None	This is the name of algorithm to use for transferring data between client and server. The algorithm should be specified in the “pointbase.ini” file on the client. The choices are: twofish, blowfish, des, des3, tea , and idea. Use <none> to specify no encryption. If the user specifies “defaultAlg” as the value then the Blowfish algorithm is used. Refer to the “Cryptography” chapter of this guide for more details on each of these algorithms.
crypto.communicationKey	Private	This is the encryption key used to encrypt data which is transferred between client and server. The key should be specified in the “pointbase.ini” file on the client. Refer to the “Cryptography” chapter of this guide for more information.
crypto.DatabaseAlgorithm	none	This is an algorithm used to encrypt data in the database. Options are BLOWFISH, DES, DES3, TEA, IDEA, TWOFISH and none. Refer to the “Cryptography” section of this guide for more information.
crypto.databaseKey	Private	The database encryption key used across the network. Refer to the “Cryptography” section of this guide for more information.
cursor.holdAcrossCommit	FALSE	Sets the default for result set holdability. See “Using Result Sets” in the *PointBase Developer’s Guide* for more information.
database.home	\pointbase\databases	The folder in which database and log files are located.

Table 1: PointBase Parameters

Key Name	**Default**	**Description**
database.pagesize	4096	The size, in bytes, of each database page. This can be any number divisible by 1024. The minimum size is 1K and the maximum size is 32K. This property is valid only when creating a new database.
debug.log	FALSE	An optional debug.log is created in the PointBase home directory with the name of <database name> + "Debug.log". To turn it on, set to TRUE. To turn off, set to FALSE. If set to FALSE, PointBase may still write only important exception information to PointBase error log which is created in the PointBase home directory with the name or <database name> + "Error.log". If set to TRUE and debug.log file reaches its maximum size, PointBase renames it to <database name> + "Debug _save.log and creates another debug.log to store any new information. If the new debug.log also reaches its maximum size, it replaces the old saved log and PointBase creates another new debug.log. At most, only two debug.log files will exist.
debug.logSize	200M	Sets the **total** debug.log file size. Minimum size is 1M, size smaller than 1M will not be taken. Possible values may have the following suffixes: M, G, or K. When the max log size is reached, file is renamed and a new log file is created. If reached again, the log is renamed same as the previous one which is overwritten.
debug.logLevel	1	Sets the debug level from 1 to 3. A larger number of debug level will dump more trace information. For example, debug level 1 dumps SQL statements, level 2 dumps locks info when lock_wait timeout occurs, and level 3 dumps setXXX values in prepared statements.
documentation.home	<install_directory>\docs\server_embedded	This parameter is set to a folder where Console looks for html help files.
fetch.blocksize	100	The number of rows to fetch in each request to the database in retrieving rows of data. This parameter affects PointBase Server Option only.

Table 1: PointBase Parameters

Key Name	***Default***	***Description***
locale.country	localeLanguageDefault	Sets the default country. Can be used in conjunction with locale.language. This defaults to the country relative to the locale.language parameter value.
locale.language	null	Sets the default language. Can be used in conjunction with locale.country. In this parameter, null is equivalent to US.
locks.maxCount	2000	The number of row level locks held on a particular table before it is converted to a table lock.
locks.timeout	60	The period of time (in seconds) the database waits for a lock to be acquired before rolling back.
log.filesize	50M	The maximum size of database log file. Once this size is reached, the database starts writing a new log file labeled with the next file number. Old log file(s) are eventually removed when they are no longer needed by the database. The lower limit for this property is 10 times the page size. The upper limit is the maximum integer value of the file size (platform-dependent). The value can be specified with an appended k/K for Kilobyte units, m/M for Megabyte as units, or g/G for Gigabyte units (for example, log.filesize=300K).
log.syncatcommit	false	When you set this parameter to true, the log file will be written to disc at the end of every transaction. When set to true, every time a commit is made (by manually committing or auto commit = true), a physical write to the log file on the disk system will be FORCED. When set to false, the return from COMMIT statement is done before ensuring the last commit log record is on the disk. If there is a system crash, it is possible that last few transactions that were perceived to be committed at run time did not make it to disk and got rolled back when the database is opened after the crash.
max.connections	0	The default 0 means unlimited connections. It is useful to set this to 1 if only 1 connection is to be used. This will disable the lock manager and a substantial performance gain will be realized as a result.

Table 1: PointBase Parameters

Key Name	***Default***	***Description***
server.port	9092	The server port number. Any valid port value may be used. This value should be set on both the client and the server “pointbase.ini” files.
sort.size	2048	The maximum size in KB for internal memory sorting. When 2048 is used, the database uses disk files to perform large sorts. The larger the value, the faster the sort (at the expense of memory). If the value is smaller, the memory footprint will be smaller. Use a value that is compatible with your environment.
SQLCaching.size	50	Specifies number of SQL caching entries in memory. When this value is set to 0 (zero) SQLCaching is disabled.
stream.checkSize	true	This ensures that the length of the stream specified in JDBC methods is checked against the length specified, for example, in methods like: setAsciiStream(int parameter, InputStream in, int length).
table.pageReserve	15	This integer value determines percentage of space reserved in the table pages that can be used by the UPDATE commands. This helps future updates of the rows in the page to fit into the same page.
transaction.isolationLevel	TRANSACTION_ READ_COMMITTED	Other values for this field are: TRANSACTION_REPEATABLE_READ, TRANSACTION_SERIALIZABLE, and TRANSACTION_READ_UNCOMMITTED.

Performance Tuning

This chapter explains how to improve the performance of PointBase. Improving the performance increases the speed of database operations, for example, processing queries and other SQL commands. By explaining specific database settings and how they affect performance or, by explaining the appropriate steps to optimize a query, this chapter can help you improve PointBase performance. Each of the following sections explain a different way to accomplish increased performance.

Optimizing Query Expressions

Query optimization consists of analyzing an SQL query expression used within SELECT, DELETE (where clause), INSERT (query expression), or UPDATE (where clause) statements to determine the best way to execute the query expression. The description of which resources to use and how to use them is known as a query plan.

Some of the major influences on the optimizer's choice of a query plan are the following:

- The SQL table(s) in the database to be accessed.
- The size of each SQL table (number of rows and columns which impacts the number of reads from physical storage).
- The indexes on the specified SQL table or tables and how they relate to the query expression.
- The size of the cache that can contain the rows of the SQL table or tables.

Indexes are very crucial to the optimizer. If a column or columns of the index are used in the where clause or as a joining column, then the optimizer can consider using the index in the costing of a given query plan. The optimizer measures the cost of using any index that exists. Indexes allow the database system to only access those rows that meet the criteria of the query expression.

For an index, the optimizer determines the number of leaf pages (bottom tier), the depth of the index and the selectivity factor. The selectivity factor is the number of unique index key column values. Additionally, the optimizer determines the cost of scanning each row of the table.

The optimizer in the PointBase is a cost-based optimizer. This means that it will look at the numerous possible query plans and will determine which one is most optimal and least expensive for a given query expression.

One of the difficulties that an optimizer encounters is that the index information can become quickly outdated. This occurs through deletions, insertions, and modifications to the actual key values of the index. The number of leaf pages, the depth and the selectivity factor all become invalid with just one or more deletions, insertions, or modifications. To solve this problem, most database systems require a database administrator to run a statistics tool to update the information on an index so that the optimizer always has current information.

The optimizer in PointBase does not require any such tool or the intervention of a database administrator. The PointBase system automatically keeps the number of leaf pages, index depth, and selectivity factor for each index consistently current. By keeping pertinent information current, the optimizer will always have correct information to determine what query plan should be employed to execute the query expression.

Execution Plan

Whenever a query is compiled, the optimizer figures out various ways the query can be executed and picks the one with the lowest cost. The cost is determined in terms of the number of I/Os needed to perform the query in addition to the CPU cost associated with evaluating the portion of the query under consideration. The important elements of an execution plan are as follows:

The Access Methods

If there are any indexes, are they used? If there are multiple indexes, which ones are used? If an index is used, does the base table need to be accessed, or is all the information needed available in the index? If only the index is accessed and not the base table, this is known as an index-only access and can improve performance dramatically.

The Join Order

This determines the order in which tables are accessed. At each step, the execution plan estimates costs and the number of rows produced.

The Join Type

This element could be a NESTED LOOP or an OUTER JOIN NESTED LOOP.

The Predicates

Where in the execution plan are the predicates put to use? The objective is to push the computation as close to the data and filter it out as quickly as possible.

PLAN Facility

To generate the execution plan for a particular query, you may use the PLAN facility by executing either of two PointBase Commander commands: EXPLAIN or SET PLANONLY ON. Before using the PLAN facility, make sure that no existing tables with the names, PLAN_TABLE and PLAN_QUERIES exist in your current schema.

If they do exist, make sure they have the correct columns by verifying them against the tables described in the next sections. If they do not have the correct columns, you must drop the PLAN_TABLE and PLAN_QUERIES tables before using the PLAN facility.

The following explains how to use the PLAN facility:

1. From PointBase Commander, enter the following: SET PLANONLY ON.
2. Compile and execute the query of interest. PointBase automatically returns the execution plan for that query.
3. Finally enter: SET PLANONLY OFF. Once this is done, you can view the PLAN_TABLE and PLAN_QUERIES tables with a SELECT statement.

The three steps explained above, can also be performed using the EXPLAIN command. Using the PointBase Commander, enter the keyword, EXPLAIN, followed by the query for which you want to create an execution plan. You may use INSERT, DELETE, UPDATE, or SELECT for the query, for example:

```
EXPLAIN select * from t1;
```

Note that, using the EXPLAIN command clears the tables, PLAN_QUERIES and PLAN_TABLE, each time you use the command. Using the EXPLAIN command is equivalent to the following commands:

```
DELETE FROM PLAN_TABLE;
DELETE FROM PLAN_QUERIES;
SET PLANONLY ON;
SELECT * FROM <table name>;
SET PLANONLY OFF;
```

PLAN_QUERIES and PLAN_TABLE

The PLAN facility keeps the execution plan information in two SQL tables: PLAN_QUERIES and PLAN_TABLE. These SQL tables can be accessed and modified with common SQL commands. These tables have the following characteristics:

- The system automatically creates these tables.
- The owner of each table is the current user.
- As the user, you must truncate the tables to remove data that it no longer needs. To remove data from these tables, use a DELETE statement.

PLAN_QUERIES

The PLAN_QUERIES table stores the queries you executed after first executing, SET PLANONLY ON. For example, the following queries are stored the PLAN_QUERIES table:

```
select * from t1,t2 where t1.c1 = t2.c1 and t1.c1>5 and t2.c1<100;
select * from t1 LEFT OUTER JOIN t2 on t1.c1=t2.c1;
select max(t1.c1) from t1,t2 group by t2.c1, t1.c1;
```

PLAN_QUERIES

Query	*Value*
1	select * from t1,t2 where t1.c1 = t2.c1 and t1.c1>5 and t2.c1<100
2	select * from t1 LEFT OUTER JOIN t2 on t1.c1=t2.c1

PLAN_QUERIES

Query	*Value*
3	`select max(t1.c1) from t1,t2 group by t2.c1, t1.c1`

PLAN_TABLE

The PLAN_TABLE table stores the execution plan for all SQL queries after executing the command, SET PLANONLY ON or EXPLAIN. Note that PointBase automatically clears this table after every EXPLAIN command. If you are using the PointBase Commander, make sure to use the SET SCREENWIDTH <value> command to view all the columns in PLAN_TABLE, for example: SET SCREENWIDTH 2000;

PLAN_TABLE

Query	*Block*	*Step*	*Operation*	*Access_ Method*	*Table_ Name*	*Index_ Name*	*Cost*	*Cum. Cost*	*Output _Rows*	*Expressions*	*Begin Key Predicates*
1	1	1	scan	indexonly scan	t1	ind1	1	1	5	<(T2.C1, constant)	=constant
1	1	2	nested loop join	table scan	t2	Null	10	11	15	=(T1.C1, T2.C1), >(T1.C1,constant)	=constant
2	1	1	none	table scan	t2	Null	5	5	20	Null	>constant
2	1	2	outer nested loop join	table scan	t1	Null	5	10	16	=(T1.C1, T2.C1)	Null
2	1	3	group by	Null	Null	Null	Null		Null	(T2.C1,T1.C1)	Null
3	1	2	scan	table scan	t3	Null	30	30	40	Null	Null
3	1	3	nested loop join	table scan	t4	Null	20	50	80	Null	Null

Optimizing MIN and MAX Functions in a Query

This section explains when the optimizer can optimize the MIN and MAX functions in a query, which is also referred to as *MIN/MAX Optimization*. Optimizing MIN or MAX functions increases the performance of a particular query that uses these functions. If a query meets certain criteria, then at best, PointBase only has to read one row instead of reading all rows that satisfy the key predicate(s). The following sections summarize the descriptions of MIN and MAX functions and explain the necessary conditions for the optimizer to choose MIN/MAX Optimization.

MAX Function

The MAX function returns the data item with the highest value for a column when applied to a column containing numeric data. If you apply the MAX function to a CHARACTER value, it returns the last value in the sorted values for that column. See the *PointBase Developer's Guide* for more information about MAX functions.

MIN Function

The MIN function returns the data item with the lowest value for a column when applied to a column containing numeric data. If you apply the MIN function to a CHARACTER value, it returns the first value in the sorted values for that column. See the *PointBase Developer's Guide* for more information about MIN functions.

MIN/MAX Optimization

For the optimizer to choose MIN/MAX Optimization, the query must meet certain criteria. There can only be one aggregate in the SELECT list: a MIN aggregate or a MAX aggregate. There can be no GROUP BY or HAVING clauses, and there can be no aggregates in the WHERE clause.

If the query meets these specified criteria, the optimizer can choose MIN/MAX Optimization to find the minimum or maximum of a column, which is either the leading column of an index or is an index column up to and including the first column after all index key columns that are covered by equality key predicates. It is also possible if there are range key predicates, but only if the query performs MIN on an ASCENDING index or MAX on a DESCENDING index.

The following are examples of MIN or MAX optimization for the given index:

```
CREATE INDEX PUBLIC.IND2 on PUBLIC.ORDER_TBL ( CUSTOMER_NUM, REP_NUM,
PRODUCT_NUM); /*Index columns are ASCENDING by default*/

SELECT MIN (CUSTOMER_NUM) /*CUSTOMER_NUM is the leading index column*/
FROM ORDER_TBL;

SELECT MIN (REP_NUM) /*REP_NUM is an index column up to and including the
first column after all index key columns that are cover by equality key
predicates*/
FROM ORDER_TBL
WHERE CUSTOMER_NUM = 1;

SELECT MIN(CUSTOMER_NUM) /*MIN on ASCENDING index column*/
FROM ORDER_TBL
WHERE CUSTOMER_NUM > 1; /*range key predicate*/
```

MAX on Ascending and MIN on Descending

In addition to the previous conditions, your query must also satisfy the following conditions, if you are performing the MIN function on a *descending* index column or, the MAX function on an *ascending* index column.

- For MIN or MAX, you cannot have any non-key predicates in the query.
- For MIN, the index column that you are querying cannot allow any NULL values.

Optimizing Count(*) in a Query

This section explains when the optimizer can optimize the count(*) function in a query, which is also referred to as the Count(*) Optimization. Optimizing the count(*) function increases the performance of a particular query that uses this function. If a query meets certain criteria, then PointBase can return it's internal statistical count of the number of rows in the table as the count. Without the optimization, PointBase must read every row in the table and count every row seen. The following sections summarize the description of the count(*) function and explains the necessary conditions for the optimizer to choose the count(*) optimization.

count(*) Function

The count(*) function returns the number of rows that satisfy the query after any where clause predicates have been applied. If there is a group by clause, then a separate count is returned that represents the number of rows in each group. See the PointBase Developer's Guide for more information about COUNT functions.

count(*) Optimization

For the optimizer to choose count(*) Optimization, the query must meet certain criteria. There can be only one table in the from clause, and only one aggregate in the select list: a count(*) aggregate. There can be no group by or having clause, and no where clause. In other words, the query must only be asking for the count of every row in the table. Under these conditions, then the internal count of rows for the table is equivalent to the query result. Pointbase will simply read the internal count and return. This will be very fast.

If the transaction isolation level is not read uncommitted, then to ensure the consistency of the count(*) result, we will attempt to get a shared table lock on the table before reading the count. If we must wait to get the lock, we will not use the optimization and will compute the count in the normal fashion. We will not wait because if the table is busy, we could wait a long time to get the table lock, perhaps longer than it would take to just compute the count by reading all the rows. The table lock will be released in the usual manner, according to the transaction isolation level.

Examples

The following are examples of the count(*) optimization:

Schema:

```
T1 (a int not null, b int not null, c  int);
T2 (a int not null, b int not null, c  int);
```

Query 1:

```
Select count(*) from t1;
```

Count(*) optimization is possible

Query 2:

```
Select count(*) from t1 where a < 100;
```

Count(*) optimization is NOT possible

Query 3:

```
Select count(*),min(a),max(a) from t1;
```

Count(*) optimization is NOT possible

Query 4:

```
Select count(*) from t1,t2;
```

Count(*) optimization is NOT possible

Query 5:

```
Select count(*) from t1 group by a;
```

Count(*) optimization is NOT possible

Tuning Database Properties

By configuring some of the database properties using the “pointbase.ini” parameters, you can enhance the performance of PointBase. The following describes the “pointbase.ini” parameters you can configure to improve performance. For a complete list and description of the “pointbase.ini” parameters, refer to "PointBase Parameters" on page 32.

Increasing the Cache Size

Some performance improvement may be seen by increasing the size of the cache, which reduces I/O. To set this size, change the value for the “pointbase.ini” parameter, “cache.size.” The default value is 2063 buffers where each buffer holds a page. For best results, the cache size should be a prime number. Increasing the size may give improved performance at the expense of memory usage.

Setting the Database Page Size

Changing the value for the parameter, “database.pagesize,” sets the default pagesize for the database at creation time. PointBase supports up to 16 different page sizes within a database for different tables, as well as for indexes, and for blob and clob columns. Table and Index pages support sizes of 1K to 32K in increments of 1K. BLOB and CLOB pages support any size in increments of 1K. Careful attention to the optimum page size for a table, column, or index can result in a performance improvement.

For example, if you always access a number of rows that require a page size less than 1K, you should set the default to 1K. If you always access a number of rows that require a page size more than 32K, you should set the default to 32K. If you always access a number of rows that require a maximum page size of 32K or a minimum page size of 1K, you should set the default page size to 4K (which is the PointBase default).

With 4K as your default page size, if you access a number of rows that require a page size between 1K and 4K, you do not waste a significant amount of space on the page. If you access a number of rows that require a page size between 4K and 32K, PointBase will create additional pages as needed in multiples of 4K.

Setting the Number of Rows Returned

By knowing the number of rows of data you need to retrieve or fetch from the database, you can configure the “fetch.blocksize” parameter to minimize network traffic and improve performance. (This applies to PointBase Server Option only). This can also be set via a JDBC statement for individual statements.

Disabling Some Logging

You can improve the performance of your system by setting the “log.forcewrite” parameter to FALSE. By setting this parameter to false, a physical write to the log file on the disk system will NOT be forced after each COMMIT. However, this can have a negative effect on the recovery process. If there is an abnormal termination of the program, it is possible that some committed transactions will be lost. In cases where recovery is not an important issue, it is safe to set this to FALSE.

Setting Memory Usage for Sorting

If you are sorting or using ORDER BY statements, the “sort.size” parameter can improve the performance of the system. This parameter is used to store rows in memory while sorting. If current memory is not enough, then a temporary file is used. Tune this parameter depending on the size of the result sets you are sorting. The larger the value, the faster the sort (at the expense of memory). If the value is smaller, the memory footprint is smaller.

Caching SQL Statements

The “SQLCaching.size” parameter configures the number of statements with the plan stored in cache memory. If you know the number of statements that your system uses repeatedly, you can configure this parameter to match that same number of statements, improving the performance of the system. If an SQL statement is stored in the cache, the compilation and plan creation are eliminated, which can save time.

Reserving Table Space for Updates

By configuring the “table.pageReserve” parameter you can reserve space in each table to improve the performance of the system every time it performs an UPDATE. This parameter specifies how much space should be reserved while inserting rows into a table. The space reserved will be used when updates increase the row length. If enough space is not available for the updated row, then a new page will be allocated for the updated row. Having enough space reduces the need for page allocation. If the table is not likely to be updated, then you can set this value to 0.

Minimizing Locking Times

This section explains how to minimize locking times in PointBase to increase performance. When you hold a lock on a table or row, it prevents another transaction to perform actions on the same table or row, which may adversely affect performance. (The action prevented depends on the type of lock held.) The following sections explain how to minimize locking times in different ways to increase PointBase performance.

Writing Data at the End of Transactions

When possible, attempt to write data to the database at the end of a transaction. Since the write locks will only be released once the transaction is committed, you can minimize the time that the write locks will be held. This is especially important for *hot-spot* tables. These are tables that have a small number of rows and that almost every transaction must update—also known as *Hot-Spot* tables.

Using READ_COMMITTED

When reading data and the transaction-isolation level is set to READ_COMMITTED, PointBase releases the lock on a row as soon it returns the row data to the user, minimizing resource usage and maximizing concurrency. After all the reads are complete, no locks are held.

Using Indexes

This section explains how to use indexes with PointBase to increase performance. Indexes allow you to increase the speed of data access to disk. Poor use of indexes may adversely affect performance. The following sections explain different ways to use indexes to improve PointBase performance.

Ordering Columns

When creating indexes, order the index columns with performance in mind. That is, the first index column should be a column for which you will most often have an equality key predicate.

If you need to sort data by particular columns (ORDER BY) or need to group rows on particular columns (GROUP BY), consider adding an index on those columns in the order by which you need the data to be ordered or grouped. The optimizer can then use the index to access the rows in the correct order—avoiding a sort.

Accessing Indexes Only

Whenever possible, consider adding enough columns to the index so that all columns necessary for queries which use the index can be satisfied from the index itself. This type of access is called "index-only" access.

Avoiding Parameters in Range Predicates

If possible, do not use parameters for range predicates on indexed columns. They make it hard for the optimizer to determine how many rows will be accessed when using that index, making it a difficult task for the optimizer to determine if using the index will be more efficient than a table scan.

Scanning Indexes

To make sure your indexes are being used, use the EXPLAIN command. As discussed in the section, "PLAN Facility" on page 38, the EXPLAIN command returns the execution plan for a particular query. You may use EXPLAIN to see if you are getting index scans. If this is not the case, and you feel you should be getting index scans, you may not have set up your indexes correctly. Make sure to re-evaluate your indexes.

Calculating Index Overhead

Although index management can improve performance, it may add overhead to INSERT and UPDATE operations. PointBase recommends calculating the correct balance so that the overhead does not exceed the benefits. (Consequently, you should create indexes on columns—only when they use the indexes in predicates—where the majority of results return a small number of rows.) Use the following formula to calculate the cost/benefit ratio:

```
# of rows selected / # of rows in table=cost-benefit
```

If the result is less than .10, then you should create indexes. If the result is greater, then you should not create an index, as the overhead will cost more than its benefits.

Selecting Multiple Rows With Different Key Values

If you need to select multiple rows with different key values, consider using an IN predicate instead of multiple OR predicates. This will be advantageous if all the OR values are close together.

If all the OR values are not close together, consider using UNION DISTINCT instead of multiple OR predicates. This will return the same result as using multiple OR predicates if the primary key column(s) are being selected.

Using the UNION Operator

Use UNION ALL versus UNION DISTINCT if there are no duplicate rows between the two parts of the union or if duplicate rows are acceptable.

PointBase™ Cryptography

Cryptography is the science of data protection. Encryption algorithms vary in their strength (or their perceived ability to protect data) and performance, but have many aspects in common. The symmetric ciphers used in the PointBase products all operate on blocks of data, using an encryption key to produce new data blocks. These data blocks may later be used to reconstitute the original data, provided the same key is used. A good algorithm, one that is said to be *strong*, has properties that make plain-text difficult to decipher. PointBase uses encryption in two places; in the database file(s), and between the client and server for network communications.

Note: Encryption is not available in the evaluation version of PointBase.

Database Encryption

In this type of cryptography, the database pages themselves are encrypted. This type works with the PointBase Embedded version and with the PointBase Embedded - Server Option. It is possible to set the database to use a supported algorithm to encrypt all the data that is being written to the disk files.

The performance overhead of each algorithm is different; however, it has been shown that none of the supported algorithms contribute to any significant degradation to the database performance.

Client and Server Communication Encryption

In this type of cryptography, a session key is generated by the client, using a zero-information protocol negotiated with the server. This session key is then used in a symmetric stream cipher, picked randomly each time a connection is made. The session key is destroyed each time a connection is dropped. From this point on, all communications between client and server are encrypted, including the database passwords and data.

Available Algorithms

There are currently six available algorithms for your PointBase software: BLOWFISH, DES, DES3, TEA, IDEA, and TWOFISH.

Algorithm	**Description**	**Key Length**
BLOWFISH	Blowfish is a complex modular algorithm that uses a session key. It is very secure and relatively fast, but its key setup is designed to be relatively slow.	Variable from 32 to 448 bits
DES	An older type of encryption algorithm that works best on hardware, but is extremely slow on software. It uses a 56-bit session key.	56 bits
DES3	DES3 is similar to the older DES algorithm, but the 56-bit session key is used three times, thereby increasing security.	56 bit key used 3 times (= 168 bits)
TEA	TEA is a cryptographic algorithm designed to minimize memory footprint while maximizing speed.	128 bits
IDEA	IDEA is considered a highly secure block algorithm. Its 128-bit key is resistant to most decryption attempts.	128 bits
TWOFISH	TWOFISH is considered the most secure block algorithm. Its 256-bit key 128-bit block cipher with 16 rounds has thus far not been broken.	256 bits

For more information on the various cryptographic algorithms, go to *http://www.kremlinencrypt.com/crypto/algorithms.html* or read *Applied Algorithms* by Bruce Schneier.

Setting Database Encryption

To set database encryption algorithms, you need to modify the “pointbase.ini” file on the server. You need to add or update the following line to include the name of the algorithm. By default the algorithm is none:

```
crypto.DatabaseAlgorithm=<algorithm_name>
```

For example:

```
crypto.DatabaseAlgorithm=tea
```

NOTE: The crypto.DatabaseAlgorithm property has no default setting.

To set the database encryption key, you need to set the "pointbase.ini" parameter crypto.databaseKey. Every PointBase encryption algorithm uses a key to encrypt data. You can use any string value for this parameter, for example:

```
crypto.databaseKey=pointbase
```

NOTE: This parameter has a private default setting if no key is specified.

For more information on updating the "pointbase.ini file," refer to the "Configuring PointBase" section.

Setting Client and Server Communication Encryption

Similar to the database encryption, PointBase offers client and server communication encryption. This database property is set only on the client machine, because the client initiates the encryption. When the client connects to the server, it passes the "pointbase.ini" parameter crypto.communication as true. This parameter is set to true by default. You can set this encryption off by setting the "pointbase.ini" parameter to false. The following is an example:

```
crypto.communication=true
```

However, if you decide to keep this encryption on, but want to change the algorithm, you can do so by setting the "pointbase.ini" parameter crypto.communicationAlgorithm. Set this parameter in the same way you set the crypto.DatabaseAlgorithm, described in "Setting Database Encryption Algorithms." The following is an example:

```
crypto.communicationAlgorithm=blowfish
```

Also similar to database encryption is the parameter crypto.communicationKey. Every PointBase encryption algorithm uses a key to encrypt data. Set this parameter the same way as the database encryption key. The following is an example:

```
crypto.communicationKey=pointbase
```

NOTE: This parameter has a private default setting if no key is specified.

Database Log Flushing

The PointBase database logs transaction information in a log file for its internal transaction management and crash recovery. This log file grows over time due to transaction information used when synchronizing data.

PointBase provides a mechanism for flushing the log files automatically. Instead of appending log information to the same file, PointBase switches to a new file with an incrementing log file number. The old log files are automatically removed if the log information is no longer needed.

Switching to a new log file can be done by either of the following three ways:

- Setting the property log.filesize.
- Programmatically invoking the `switchLogFile()` method in com.pointbase.jdbc.jdbcConnection object (for PointBase Embedded) or com.pointbase.net.netJDBCConnection object (for PointBase Embedded - Server Option).
- Issuing a SWITCHLOGFILE command from PointBase Commander.

Since old log files are deleted automatically, you should not explicitly delete them.

How to Flush the Log Setting the log.filesize

The log.filesize property represents the maximum size of a database log file (default=50MB). This property is set in the “pointbase.ini” file. Once this size is reached, the database starts writing into a new log file labeled with the next file number. Old log file(s) are eventually removed when they are no longer needed by the database. The lower limit for this property is 10 times the page size. The upper limit is the maximum integer value of the file size, which is platform-dependent. The value can be specified with an appended k/K for Kilobytes, m/M for Megabytes, or g/G for Gigabytes (for example, log.filesize=300K).

How to Flush the Log Using PointBase Commander

Database log flushing can also be done by issuing a SWITCHLOGFILE command from PointBase Commander. The SWITCHLOGFILE command forces the database to switch to a new log file.

Syntax:

```
SWITCHLOGFILE
```

(Single word, no parameters.)

This is an internal command understood only by PointBase Commander.

PointBase™ Internationalization

PointBase supports the Unicode standard for facilitating internationalization of applications. Unicode is an international character set that supports over 650 of the world's languages, such as Japanese, Chinese, Russian, French, and German. The Unicode standard uses a 16-bit set instead of 8-bit code sets used by other character sets.

This expansion provides codes for more than 65,000 characters, an increase of over 200 times the capacity allowed by 8-bit code sets (256 characters). To keep character coding simple and efficient, the Unicode standard assigns each character a unique 16-bit value and does not use complex modes or escape codes to specify modified characters or special cases. This simplicity and efficiency makes it easy for computers and software to handle Unicode-encoded text files.

Unicode allows developers to create a single version of an application and deploy it anywhere without requiring any modification to the code. The same Unicode application will work with data in any language. In addition, the application will run on different language versions of the operating system; for example, it can be deployed on a European language version of Microsoft Windows NT 4.x as well as on a Japanese or Chinese version without any modification. This results in a significant reduction in development, testing, and maintenance costs.

The following sections further explain Unicode.

- "What Types of Businesses are Likely to Use Unicode?"
- "Are There any Limitations to Unicode?"
- "Implementation"
- "Restrictions"

What Types of Businesses are Likely to Use Unicode?

The following businesses are most likely to use Unicode:

- Multinational companies that want to mix data from many offices and languages in one database server.
- Software developers who want to create one version of an application and send it to multiple countries without making any engineering changes.
- In-house MIS teams in banks and insurance companies.
- Libraries and museums which require that some fields contain text excerpts or names in many different languages. Other users of Unicode are software companies providing software to these institutions.
- Government departments (for example, immigration or intelligence) or non-profit organizations that must keep records in many languages.

Are There any Limitations to Unicode?

Unicode presents the following limitations:

- Unicode is very powerful but it does not do everything. Unicode is intentionally designed not to handle complex text-based operations such as sorting, hyphenation, and line breaks, or to include font or display information. For these operations, Unicode relies on the application or on the operating system.
- Unicode does *not* translate the text in applications into localized versions.
- Unicode does *not* translate data from one language to another.

Implementation

Unicode characters are multibyte letters and numbers that are used in most of the world’s alphabets. Three areas are important for Unicode support:

1. “CHAR/CLOB handling”
2. “Expression package”
3. “Sort package”

CHAR/CLOB handling

PointBase makes use of a standard string-encoding algorithm called UTF-8. UTF-8 is a 1-to-3-byte, variable-width encoding of Unicode. Different characters are one, two, or three bytes in length. For example, ASCII characters (A-Z, a-z, 0-9, and so on) are encoded as one-byte characters. Asian ideographs are encoded as three-byte characters. Accented European characters are encoded as two-byte characters. UTF-8 is the format, which is usually supported by database vendors and is a more compact storage format if the data is primarily ASCII (English) data.

Changing all string data to use UTF-8 avoids truncating Unicode characters when stored. There is a slight overhead when using this encoding algorithm.

Expression package

The expression package is responsible for the database expressions that manipulate strings. These expressions are:

- < (less than), <= (less than or equals too)
- > (greater than) >= (greater than or equals too)
- = (equals), !=, <> (not equals)
- Like (pattern matching)
- Upper/Lower (case conversion)
- Cast (from/to string types)

The source to an expression may be a literal, column or the result of another expression (implicitly or explicitly through a marker variable). These are all Unicode capable. For the expression LIKE the `%` wild card character will match zero or more Unicode characters. The `_` (underscore) wild card matches exactly one Unicode character. Character sets that support variants (like German, where the "o" has many variants and French, where "e" may be accented) require an exact match. In a cast expression where the source is a number and the target is a `String`, there will not be any locale specific formatting, as for example, in France, where the comma is used to delimit the thousands in a number.

Sort package

Databases make frequent use of a sort package. Indexes are inherently sorted, and ORDER BY and GROUP BY statements also require sorting. When characters are stored using the American ASCII character set, strings can be very efficiently sorted. Nothing more sophisticated than a byte-for-byte comparison is required. This method does have its limitations, but is generally used because of its inherent performance. One such limitation is that ASCII represents the uppercase and lowercase alphabets separately and therefore an A may sort higher (or lower depending on the sort direction) than a b. With Unicode, the order of data sort is changed. Byte-for-byte comparisons cannot be performed. Each `String` represented on a database page must now be converted back to its `String` representation. From the collation sequence (that is a product of the tables country and language) a collation key is generated. This key is then used to perform a comparison with another `String's` key. The collation key is capable of knowing, for any particular locale, the way in which `Strings` compare with one another.

PointBase supports Internationalization at three different levels:

- Database: all schemas of the database.
- Schema: all tables and indexes of a schema.
- Table: all index and column/row values of a table.

NOTE: CLOB columns behave exactly as CHAR columns in all respects with regard to Unicode. They may be used in expressions, may be sorted and may be stored/retrieved as Unicode streams.

Restrictions

Unicode presents the following restrictions:

- The database will accept parameters that are Unicode `Strings`; however, a SQL literal must contain only characters that can be entered with a keyboard. There is no ability, as in Java, to enter a Unicode character using a delimited form (for example \u0131).
- Variants on a single character are considered different in expressions (accented letters).
- There is a performance impact for Unicode tables in index creation and expressions that require `String` columns to be operated on. This is a result of converting UTF-8 encoded byte sequences to `Strings`.
- It is not possible to join two tables that have different collation sequences (the name of the combination of language and country codes) on any CHAR/CLOB column.

Using PointBase™ With IDEs

The following instructions explain how to integrate Sun™ ONE Studio, Borland JBuilder, IBM VisualAge, and WebGain Visual Cafe with PointBase. Note that several versions of the following IDEs exist and may cause the setup to vary slightly.

NOTE: The following IDEs bundle PointBase: WebGain Visual Cafe, Sun™ ONE Studio, and Motorola (Metrowerks) Code Warrior.

Sun™ ONE Studio

This section explains how to integrate PointBase with Sun™ ONE Studio IDE.

1. Select Add `jar` from the Tools menu.
2. Use the Browse button to navigate to your PointBase `jar` files.
   Once you have selected your `jar` file, it displays under File Systems in the Explorer window.

   Click the Project Defaults tab in the Explorer window.
3. Click the Add existing button.
4. Select Objects window.
5. Select a specific `jar` and click Ok.

Borland JBuilder

This section provides information on how to integrate PointBase with Borland JBuilder 3.5 (Java) IDE.

To use PointBase with Borland JBuilder 3.5 (Java) IDE, you first need to create a new Java project following the standard documentation from Borland. Once your project is created, follow these steps to add PointBase:

1. Select Project Properties from the Project menu.
2. Select the Required Libraries tab and click the Add button on the dialog box.
3. Click the New button.
4. Enter POINTBASE in the Name field.
5. Click the Add button on the Class tab.
6. Browse to the location of your PointBase `jar` file (for example, `\pointbase\lib\pbembedded44.jar`)
7. Select the `jar` file and Click Ok.
   Repeat this step for other `jar` files, for example, `pbclientxx.jar,` where *xx* is the release number.
8. Click OK on the Class tab.
   In the dialog box, “Select one or more libraries,” POINTBASE should now appear in the list of possible libraries.
9. Select POINTBASE and click Ok.
   POINTBASE now appears in the Required Libraries tab of the Project Properties dialog box.
10. Select Ok. You are now ready to write JDBC projects that access PointBase.

If you wish to make PointBase available to ALL default projects using JBuilder, follow the previous procedure with the following change to step 1:

1. Select Default Project Properties from the Project menu.

IBM VisualAge

This section explains how to integrate PointBase with VisualAge for Java.

1. Go to the File menu.
2. Select the Quick Start menu item in VisualAge.
3. Select Create Project from the Basic list and press Ok.
4. Create a Project named PointBase and press Finish.
   The WorkBench now contains a project named PointBase.
5. Right-click on the project named PointBase in the Workbench window and select Import from the list
6. Select `jar` file and press Next.
7. Use Browse to locate the "p`bembedded`xx.jar" file in the lib subfolder of the `\pointbase\` directory (where *xx* represents the release number, for example 44).
8. Select the `class` and the resource check boxes and press Finish.

WebGain Visual Cafe 4.1

This section explains how to integrate PointBase with Web Gain Visual Cafe 4.1.

1. Select New Project from the Visual Cafe File menu.
2. Select the Data Bound Project wizard and click Ok.

   This integrates PointBase classes with Visual Cafe automatically. For more information about using Visual Cafe, please refer to the WebGain documentation.

If you are integrating a different PointBase database, other than the database bundled with Visual Cafe, perform the following:

1. Select New Project from the Visual Cafe File menu.
2. Select Create Empty Project and click Ok.
3. Select Options from the Project menu.
4. Select the Directories tab.
5. Select Input class files from the "Show directories for" drop down field.
6. Click the New button located directly underneath the "Show directories for" drop down field.
7. Enter folder name with the complete directory path to the
   `jar` file, in the specified field.
   or
   Click the File button or the button directly to the right of the specified field.
8. Navigate to the PointBase classes in the PointBase `lib` directory.
9. Select your PointBase `jar` file and click Ok. (Accept the defaults.)

Appendix A: System Tables

This chapter describes the PointBase system tables. The PointBase RDBMS stores information about data structures in system tables. You cannot modify these system tables, because they are automatically updated as a result of SQL operations by the database system.

However, these system tables can be queried using standard SQL SELECT statements. For example, to view the components of table SYSTABLES, type the following:

```
select tablename from systables;
```

The following is a list of the PointBase system tables:

- "SYSDATABASES" on page 62
- "SYSTABLES" on page 62
- "SYSVIEWS" on page 63
- "SYSVIEWTABLES" on page 63
- "SYSCOLUMNS" on page 64
- "SYSINDEXES" on page 64
- "SYSINDEXKEYS" on page 65
- "SYSPAGESIZEMAP" on page 65
- "SYSSCHEMATA" on page 66
- "SYSUSERS" on page 66
- "SYSROLES" on page 66
- "SYSSQLDATATYPES" on page 67
- "SYSTABLECONSTRAINTS" on page 67
- "SYSKEYCONSTRAINTCOLUMNS" on page 67
- "SYSREFERENTIALCONSTRAINTS" on page 68
- "SYSCHECKCONSTRAINTS" on page 68
- "SYSTRIGGERS" on page 69
- "SYSTRIGGERCOLUMNS" on page 70
- "SYSROUTINES" on page 70
- "SYSPARAMETERS" on page 71
- "SYSTABLEPRIVILEGES" on page 71
- "SYSCOLUMNPRIVILEGES" on page 72
- "SYSROLEPRIVILEGES" on page 72
- "SYSROUTINEPRIVILEGES" on page 72
- "SYSTRIGGERROUTINEDEPEND" on page 73
- "SYSCONSTRAINTROUTINEDEPEND" on page 73
- "SYSSQLSTATEMENTS" on page 73

SYSDATABASES

This table contains information about the PointBase database.

Column Name	***Data Type***	***Description***
DatabaseName	varchar(128) not null	The name of the database
DatabaseId	integer not null	The database ID number
OwnerName	varchar(128) not null	The username of the database owner
Creation	timestamp not null	The time that the database was created
Location	varchar (900) not null	The location of the database file

SYSTABLES

This table lists attributes of the tables in a PointBase database.

Column Name	***Data Type***	***Description***
SchemaId	Integer	System generated id of the schema
TableName	Varchar(128)	The name of the table
TableId	Integer	System generated id of the table
TableType	Integer	1: Base 2: System
TableCreation	Timestamp	Creation time of the table
TableLastModified	Timestamp	Time of last table modification
TableFirstPage	Integer	First page to the table
PageSize	Integer	The size of the table, in kilobytes. “0” in this field indicates the table uses either the default pagesize which is 4 KB or the pagesize specified in the `pointbase.ini` file. The maximum pagesize is 32 KB. All pagesizes should be a multiple of 1 KB.
LOBPageSize	Integer	The size of the LOB table, in kilobytes. “0” in this field indicates the table uses either the default pagesize which is 4 KB or the pagesize specified in the `pointbase.ini` file. There is no maximum lobpagesize. All lobpagesizes should be a multiple of 1 KB.
ReplicationStatus	Boolean	For internal use only

Column Name	**Data Type**	**Description**
CommitBehavior	Short	1: on commit delete rows 2: on commit preserve rows
Country	Char(2)	The country code value contained in the country and language code property. It defaults to US ASCII. If no value is specified the default is US.
Language	Char(2)	The language code value contained in the country and language code property. It defaults to US ASCII. If no value is specified the default is US.
TableOrgType	Smallint	1: Entry Sequenced (default) 2: Key Sequenced

SYSVIEWS

This table stores the definitions of each view.

Column Name	**Data Type**	**Description**
SchemaId	Integer	System generated ID of the schema
ViewName	Varchar(128)	The name of the view
ViewId	Integer	System generated ID of the view
IsUpdatable	Boolean	TRUE if view is updateable, FALSE otherwise
CheckOption	Char(1)	N: Not specified, C: Specified WITH CASCADED CHECK OPTION, L: Specified WITH LOCAL CHECK OPTION Only applies to updateable views
ViewDefinition	Varchar(3958)	Stores the view text: the SELECT statement that defines the view

SYSVIEWTABLES

This table lists the table id of all tables referenced in the view

Column Name	**Data Type**	**Description**
ViewSchemaId	Integer	System generated ID of the view’s schema
ViewId	Integer	System generated ID of the view

Column Name	Data Type	Description
TableSchemaId	Integer	System generated ID of the referenced table's schema
TableId	Integer	System generated ID of the referenced table
TableColumnId	Integer	System generated ID of the referenced table's column

SYSCOLUMNS

This table stores the definitions of columns in a PointBase database.

Column Name	Data Type	Description
TableId	Integer	System generated id of the table
ColumnName	Varchar(128)	The name of the column
ColumnId	Integer	System generated id of the column
OrdinalPosition	Integer	Position number of the column within the table
ColumnType	Integer	SQL Data Type value
ColumnLength	Integer	Data Length/Precision (maximum)
ColumnScale	Integer	Length of the scale
IsNullable	Boolean	True: can accept SQL NULL values, False: cannot accept SQL NULL values
ColumnDefault	Varchar(900)	Default value for the column

SYSINDEXES

This table describes attributes of indices.

Column Name	Data Type	Description
TableId	Integer	System generated Id of the table
IndexName	Varchar(128)	Name of the index
IndexId	Integer	System generated Id of the index
IndexSchemaId	Integer	Represents the Schema ID of the Schema with which the index is associated.
IndexFirstPage	Integer	First page id of the index

Column Name	Data Type	Description
IndexType	Integer	Index key uniqueness. 1: primary, 2: unique, 4: duplicate keys allowed
IndexOrgType	Integer	1: B-tree 2: Clustered Btree
IndexPageSize	Integer	The size of the table index, in kilobytes. “0” in this field indicates the table uses either the default pagesize which is 4 KB or the pagesize specified in the `pointbase.ini` file. The maximum indexpagesize is 32 KB. All indexpagesizes should be a multiple of 1 KB.
IndexColList	Varchar(10)	For internal use only
IndexCreation	Timestamp	Creation time of the index

SYSINDEXKEYS

This table lists attributes of index keys.

Column Name	Data Type	Description
TableId	Integer	System generated id of the table
IndexId	Integer	System generated id of the index
ColumnId	Integer	System generated id of the column
SortDirection	Boolean	True: Ascending, False: Descending
OrdinalPosition	Integer	Position of the column within the index

SYSPAGESIZEMAP

This table stores the definitions of page sizes in a PointBase database.

Column Name	Data Type	Description
PagesizeID	Integer	The ID number of the page size
Pagesize	Integer	The size of the page in kilobytes

SYSSCHEMATA

This table contains information on schemas in a catalog.

Column Name	***Data Type***	***Description***
SchemaName	Varchar(128)	The name of the schema
SchemaOwnerId	Integer	System generated id of the owner
SchemaId	Integer	System generated id of the schema
SchemaCreation	Timestamp	Schema creation time
SQLPath	Varchar(900)	The SQL path of the schema
Country	Char(2)	The country code value contained in the country and language code property. It defaults to US ASCII. If no value is specified the default is US.
Language	Char(2)	The language code value contained in the country and language code property. It defaults to US ASCIl. If no value is specified the default is US.

SYSUSERS

This table lists user information for a database.

Column Name	***Data Type***	***Description***
UserName	Varchar(128)	Name of the user/owner
UserId	Integer	System generated id of the user/owner
Password	Varchar(128)	Password for the user/owner
Creation	Timestamp	Time of creation of the user/owner
DefaultPath	Varchar(900)	Default schema path of the user/owner
Default RoleName	Varchar(128)	The user’s default role name

SYSROLES

This table lists roles information for the database.

Column Name	***Data Type***	***Description***
RoleName	Varchar(128)	Name of the role
RoleId	Integer	System generated id of the role

SYSSQLDATATYPES

This table contains attributes of the data types listed in a schema.

Column Name	***Data Type***	***Description***
SchemaId	Integer	System generated id of the schema
SQLType	Integer	SQL standard value of the data type
Name	Varchar(30)	SQL standard name of the data type
Length	Integer	Maximum length or precision of the data type
Scale	Smallint	Scale value of the data type
Creation	Timestamp	Time of creation of the data type

SYSTABLECONSTRAINTS

This table lists the various types of constraints within a schema.

Column Name	***Data Type***	***Description***
SchemaId	Integer	System generated id of the schema
TableId	Integer	System generated id of the table
ConstraintName	Varchar(128)	Name of the constraint. The name is either specified by the user/owner or by the system
ConstraintId	Integer	System generated id of the constraint
Type	Character(1)	Type of constraint: C: Check, F: Foreign Key, P: Primary Key
IsDeferrable	Boolean	T: Deferrable, F: Immediate. Only Immediate is allowed
IsInitiallyDeferred	Boolean	F: Immediate only
ColumnCount	Smallint	Number of columns in the constraint
Creation	Timestamp	Time of creation of the constraint

SYSKEYCONSTRAINTCOLUMNS

This table lists attributes of a constraint within a schema.

Column Name	***Data Type***	***Description***
ConstraintSchemaId	Integer	System generated id of the schema of the constraint
ConstraintTableId	Integer	System generated id of the table of the constraint

Column Name	Data Type	Description
ConstraintId	Integer	System generated id of the constraint
ColumnId	Integer	System generated id of the column
OrdinalPosition	Smallint	Position of the column

SYSREFERENTIALCONSTRAINTS

This table lists relationship attributes for a referenced table.

Column Name	Data Type	Description
ConstraintSchemaId	Integer	System generated id of the schema
ConstraintTableId	Integer	System generated id of the table
ConstraintId	Integer	System generated id of the constraint
ConstraintIndexId	Integer	System generated id of the index
ReferenceSchemaId	Integer	System generated id of the schema where the referenced table exists
ReferenceTableId	Integer	System generated id of the table where the referenced table exists
ReferenceConstraintId	Integer	System generated id of the primary key of the table where the referenced table exists
ReferenceColumnCount	Smallint	Number of columns in the primary key
MatchOption	Boolean	T: Full, F: Partial
UpdateRule	Smallint	1: No Action, 2: Cascade, 3:Set Null, 4: Set Default, 5:Restrict
DeleteRule	Smallint	1: No Action, 2: Cascade, 3:Set Null, 4: Set Default, 5:Restrict

SYSCHECKCONSTRAINTS

This table lists attributes of a constraint that restricts the contents of a specific system table.

Column Name	Data Type	Description
SchemaId	Integer	System generated id of the schema
TableId	Integer	System generated id of the table
ConstraintId	Integer	System generated id of the constraint
CheckClause	Varchar(900)	Text of the check constraint

SYSTRIGGERS

This table lists triggers for a specific schema.

Column Name	***Data Type***	***Description***
TableId	Integer	System generated id of the table referenced by the trigger
Seqno	Integer	Trigger sequence number in the table
TriggerId	Integer	System generated id of the trigger
TableSchemaId	Integer	System generated id of the schema that contains the table referenced by the trigger
TriggerSchemaId	Integer	System generated id of the schema that contains the trigger
TriggerName	String	Name of the trigger
Event	String	D: Delete, I: Insert, or U: Update
Time	String	A: After or B: Before
Granularity	String	R: Row or S: Statement
IsColumnListImplicit	Boolean	T: column list is implicit, F: a column list was specified
IsAtomic	Boolean	T: true, the body of the trigger is atomic
OldRowValue	String	Old row correlation value
NewRowValue	String	New row correlation value
OldTableValue	String	Old table correlation value
NewTableValue	String	New table correlation value
Creation	Timestamp	Time of creation of the trigger
SearchCondition	String	Search Condition text of the trigger

SYSTRIGGERCOLUMNS

This table lists attributes of a trigger within a schema.

Column Name	***Data Type***	***Description***
TableId	Integer	System generated id of the table of the trigger
TriggerId	Integer	System generated id of the trigger
ColumnId	Integer	System generated id of the column of the table that the trigger references

SYSROUTINES

This table lists routines and their components.

Column Name	***Data Type***	***Description***
SchemaId	Integer	System generated id of the schema that contains the routine
RoutineName	Varchar(128)	Name of the routine
RoutineId	Integer	System generated id of the routine
RoutineType	Integer	Type of routine. 86: Function, 150: Procedure
Language	Integer	237: SQL, 246: Java
SpecificSchemaId	Integer	System generated id of the schema that contains the routine
SpecificName	Varchar(128)	Specific name used to reference the routine
IsDeterministic	Boolean	T: True, F: False
DataAccess	Integer	125: No SQL, 238: Contains SQL, 239: Reads SQL, 240: Modifies SQL
ExternalSchemaId	Integer	System generated id of the schema that contains the external routine
ExternalName	Varchar(128)	Name of the Java class that contains the external routine
ParameterStyle	Integer	237: SQL
ReturnType	Integer	SQL data type value of the value being returned
Creation	Timestamp	Time of creation of the routine
Reentrant	Boolean	T: True F: False

SYSPARAMETERS

This table lists various parameters that affect a routine.

Column Name	Data Type	Description
SchemaId	Integer	System generated id of the schema
RoutineId	Integer	System generated id of the routine that contains the parameter
ParameterName	Varchar(128)	Name of the parameter
ParameterType	Integer	PARSE_TYPE: PARAMETER: 232 (Input, Output, Input/Output parameter) PARSE_TYPE: PARAMETER: 234 (Return value) PARSE_TYPE: PARAMETER: 26 (Original return value before casting)
OrdinalPosition	Integer	Position of the parameter in the list of parameters of the routine
ParameterMode	Integer	1: In, 2: Out, 3: InOut
ParameterCode	Integer	SQL data type value of the parameter
ParameterLength	Integer	Length or precision of the parameter
ParameterScale	Integer	Scale of the parameter

SYSTABLEPRIVILEGES

This table lists ownership privileges on a table.

Column Name	Data Type	Description
SchemaId	Integer	System generated id of the schema
TableId	Integer	System generated id of the table
PrivilegeType	Character(1)	S: select, U: update, I: insert, D: delete, T: trigger, R: Reference, A: All
GranteeId	Integer	System generated id of the grantee
GrantorId	Integer	System generated id of the grantor
IsGrantable	Boolean	T: can grant privilege to others, F: cannot grant privilege to others

SYSCOLUMNPRIVILEGES

This table lists attributes of an individual column for a table.

Column Name	***Data Type***	***Description***
SchemaId	Integer	System generated id of the schema
TableId	Integer	System generated id of the table
ColumnId	Integer	System generated id of the column
PrivilegeType	Character(1)	S: select, U: update, I: insert, T: trigger, R: reference
GranteeId	Integer	System generated id of the grantee
GrantorId	Integer	System generated id of the grantor
IsGrantable	Boolean	T: can grant privilege to others, F: cannot grant privilege to others

SYSROLEPRIVILEGES

This table lists privileges for a specific role.

Column Name	***Data Type***	***Description***
GranteeId	Integer	System generated id of the grantee
RoleId	Integer	System generated id of the role
GrantorId	Integer	System generated id of the grantor
IsGrantable	Boolean	T: can grant privilege to others, F: cannot grant privilege to others

SYSROUTINEPRIVILEGES

This table lists privileges for a specific routine.

Column Name	***Data Type***	***Description***
SchemaId	Integer	System generated id of the schema
RoutineId	Integer	System generated id of the routine
PrivilegeType	Character(1)	E: Execute
GranteeId	Integer	System generated id of the grantee
GrantorId	Integer	System generated id of the grantor
IsGrantable	Boolean	T: can grant privilege to others, F: cannot grant privilege to others

SYSTRIGGERROUTINEDEPEND

This table lists the dependencies on a particular routine(s) for a specific trigger.

Column Name	***Data Type***	***Description***
RoutineId	Integer	Id of the routine
TriggerId	Integer	Id of the trigger

SYSCONSTRAINTROUTINEDEPEND

This table lists the dependencies on a particular routine(s) for a specific constraint.

Column Name	***Data Type***	***Description***
ConstraintSchemaId	Integer	Id of the schema of the constraint
ConstraintId	Integer	Id of the constraint
RoutineSchemaId	Integer	Id of the schema of the routine
RoutineId	Integer	Id of the routine

SYSSQLSTATEMENTS

This table lists the type of SQL statements that can be used.

Column Name	***Data Type***	***Description***
TRID	Integer	System generated id of the routine or trigger
Seqno	Integer	Sequence number of the statement
StatementType	Byte	1: Call, 2: Routine invocation, 3: Return
Type	String	F: Function, P: Procedure, or T: Trigger
SQLText	String	Text of the SQL statement

Appendix B: Error Messages

The following table describes PointBase error messages.

NOTE: In the message column, the numbers enclosed by braces represent parameter strings based on the context of the error.

PointBase message code	***SQL state value***	***PointBase message***
0	00000	Successful completion
1	02000	No data found.
1001	0100A	Warning -- query expression {0} too long for information schema.
1002	0100B	Warning -- default value {0} too long for information schema.
1003	0100C	Warning -- dynamic result sets returned.
1004	0100D	Warning -- additional result sets returned.
1005	0100E	Warning -- attempted to return too many result sets.
1006	01001	Warning -- cursor conflict operation. {0}.
1007	01002	Warning -- disconnect error. {0}.
1008	01003	Null value eliminated in set function.
1009	01004	String data, right truncation.
1010	01006	Privilege not revoked. {0}.
1011	01007	Privilege not granted. {0}.
1012	01008	Implicit zero-bit padding.
1013	01009	Warning -- search condition {0} too long for information schema.
1014	01H00	External routine warning. {0}.

PointBase message code	***SQL state value***	***PointBase message***
1015	01H01	ORDER BY expression not found in SELECT list.
1016	01H02	There has been a mismatch in the number of assignment terms found during type checking.
1017	01H03	To take advantage of bug fixes related to Indexes--Unload your tables. Create a new database and load them again.
2001	08001	SQL-client unable to establish SQL-connection. {0}
2002	08002	Connection name in use. {0}.
2003	08003	Connection does not exist. {0}.
2004	08004	SQL-server rejected establishment of SQL-connection. {0}.
2005	08006	Connection failure. {0}.
2006	08007	Transaction resolution unknown. {0}.
2007	2E000	Invalid connection name. {0}.
2008	ZA000	Cannot find the JDBC driver "{0}". Driver needs to be registered.
2009	ZA001	Error occurred while instantiating the JDBC driver "{0}"
2010	ZA002	No valid connection handle exists. Please reestablish a connection handle through JDBC.
2011	ZA003	The JDBC feature {0} is not supported.
2012	ZA004	A JDBC operation of {0} failed.
2013	ZA005	Please note, the PointBase software you are currently using was made available to you on a 30-day trial basis. We appreciate your interest in evaluating our software for use within your applications. The evaluation period for this software has expired. To continue using this software you will need to either purchase a full production copy from our web site at http://www.pointbase.com/products/purchase or contact PointBase sales within U.S. and Canada at 1-877-238-8798. (Intl: 1-650-230-7200 touch 2).

PointBase message code	*SQL state value*	*PointBase message*
2014	ZA006	Please note, the PointBase software you are currently using was made available to you on a 30-day trial basis. We appreciate your interest in evaluating our software for use within your applications. The evaluation period for this software will expire in {0} days. To continue using this software you will need to either purchase a full production copy from our web site at http://www.pointbase.com/products/purchase or contact PointBase sales within U.S. and Canada at 1-877-238-8798. (Intl: 1-650-230-7200 touch 2).
2015	ZA007	The system has detected that another process on this computer currently has the database open. If you are certain this is a mistake, please wait a few minutes and try again. If you are still unable to connect, you may delete the lock file, which is located in the database home directory with the same name as the database and the extension "`.lck`".
10001	30000	Invalid SQL statement at position {0}.
10002	ZB001	Double quoted string not ended at position {0}.
10003	ZB002	String not ended at position {0}.
10004	ZB003	An unexpected character was found at position {0}.
10005	ZB004	An unexpected token was found at position {0}.
10006	ZB005	Expected to find "{0}" instead found "{1}" at position {2}.
10007	ZB006	Missing query expression at position {0}
10008	ZB007	Only constant expressions allowed here at position {0}.
10009	ZB007	Missing relational operator at position {0}.
10010	ZB009	Data type {0} is not a valid SQL data type supported by PointBase. Please use one of the allowable types.
10011	ZB010	LOB size of {0}{1} is larger than 4 gigabytes which is the maximum length supported for LOB data types.
10012	ZB011	{0} is not a valid DEFAULT clause value. Valid values are NULL, constants, and datetime functions.
10013	ZB012	DEFAULT VALUE was found, but other expressions exist. This is invalid syntax. See the Reference Manual.
10014	ZB013	The format {0} is not a valid BLOB format. The format must be of the order: X'<hexit><hexit>...' where <hexit> is a digit (0..9) or A, a, B, b, C, c, D, d, E, e, F, or f.
10015	ZB014	The LIKE operator is only allowed for Character data.

PointBase message code	***SQL state value***	***PointBase message***
10016	ZB015	The grant object type: {0} is not a valid object type to grant privileges to. Valid types are TABLE, FUNCTION, METHOD, or PROCEDURE.
10017	ZB016	The grant privilege: {0} is not a valid privilege type. Valid types are: SELECT, DELETE, INSERT, UPDATE, REFERENCES, TRIGGER and EXECUTE.
10018	ZB017	The revoke object type: {0} is not a valid object type to revoke privileges to. Valid types are TABLE, FUNCTION, METHOD, or PROCEDURE.
10019	ZB018	The revoke privilege: {0} is not a valid privilege type. Valid types are: SELECT, DELETE, INSERT, UPDATE, REFERENCES, TRIGGER and EXECUTE.
10020	ZB019	It is not valid to have a TABLE trigger with an event time of BEFORE. Only ROW triggers can have an event time of BEFORE.
10021	ZB020	An AFTER DELETE statement trigger can specify OLD TABLE alias name only.
10022	ZB021	A BEFORE DELETE trigger can specify an OLD ROW alias name only.
10023	ZB022	A BEFORE INSERT trigger can specify a NEW ROW alias name only.
10024	ZB023	An old values alias is equivalent to the new values alias. Alias names must be unique per trigger definition. Please change one of the alias names.
10025	ZB022	The set parameter {0} is invalid.
10026	ZB025	Expected to find a referential action. Instead found {0}. The referential action must be one of the following: CASCADE, SET NULL, SET DEFAULT, RESTRICT, NO ACTION.
10027	ZB026	The scale {0} is larger than the precision {1}. Decimal and Numeric data types require the scale to be less than or equal to the precision.
10028	ZB027	LOB size of {0}{1} is larger than 4 gigabytes which is the maximum length supported for LOB data types.
10029	ZB028	New Pagesize {0}KB cannot be supported. The new pagesize exceeds the limit of {1} different pagesizes.
10030	ZB029	Trying to redefine {0} pagesize.
10031	ZB030	The size of {0} is smaller than the allowed minimal size of one.
10032	ZB031	Table or Index pagesize {0}KB exceeds the 32KB limit.

PointBase message code	***SQL state value***	***PointBase message***
10033	ZB032	A NEW ROW or OLD ROW alias name was specified for a statement trigger. Statement triggers can only specify OLD TABLE or NEW TABLE alias names.
10034	ZB033	An AFTER INSERT statement trigger can specify a NEW TABLE alias name only.
10035	ZB034	Only a Routine (Function or Procedure) Invocation is allowed at this point {0}.
10036	ZB035	The Trigger alias name {0} was used in the body of a trigger but was not specified in the referencing clause.
10037	ZB036	The trigger {0} was defined on table {1}.{2}. This table is not defined.
10038	ZB037	A BEFORE UPDATE trigger can specify NEW ROW or OLD ROW alias names only.
10039	ZB038	An AFTER UPDATE statement trigger can specify OLD TABLE or NEW TABLE alias names only.
10040	ZB039	The Trigger alias {0} has previously been specified.
10041	ZB040	An AFTER DELETE row trigger can specify OLD ROW or OLD TABLE alias name only.
10042	ZB041	An AFTER INSERT row trigger can specify NEW ROW or NEW TABLE alias names only.
10043	ZB042	An invalid parameter mode was specified for a parameter of an SQL Function which requires all parameter modes to be IN.
10044	ZB043	Attempted to add a constraint to a column that cannot be supported.
15000	ZD000	System Catalog error {0}.
15001	42000	Access rule violation. User {0} does not have {1} privilege on object {2}.
15002	2B000	Dependent privilege descriptors still exists on SQL object {0}.
15003	2B000	Column reference "{0}" ambiguous at position {1}.
15004	2B000	Column "{0}" not found in table at position {1}.
15005	2B000	Table or correlation name "{0}" specified twice at position {1}.
15006	2B000	Invalid table name "{0}" specified at position {1}.
15007	2B000	Invalid ORDER BY number "{0}" specified at position {1}.

PointBase message code	*SQL state value*	*PointBase message*
15008	2B000	ORDER BY expression not found in SELECT list.
15009	ZD001	Cannot find the table "{0}.{1}".
15010	ZD002	Cannot find the index "{2}" for the table "{0}.{1}".
15011	ZD009	The table "{0}.{1}" already exists.
15012	ZD004	The index "{2}" for the table "{0}".{1} already exists.
15013	ZD005	Cannot find the schema "{0}" in the system catalog.
15014	ZD006	Column "{2}" of the table "{0}.{1}" is too long. Maximum allowed length is {3} for this table pagesize. Recreate the database with a larger page size.
15015	ZD013	Multiple Primary Keys were defined for the table. Only one Primary Key is allowed.
15016	ZD014	The Key Column "{0}" did not specify NOT NULL. Key Columns do not allow null values.
15017	ZD015	The Foreign Key {0} does not map to a primary key in table {1}. Each foreign key must reference a table that has a primary key made up of the same columns as specified in the reference list.
15018	ZD016	Column "{0}" is not defined in table "{1}.{2}";
15019	ZD017	Duplicate Column "{2}" found in the table "{0}.{1}".
15020	ZD018	Column "{0}" in the select list is not a grouping column.
15021	ZD019	Cannot find the database "{0}" in system table SYSDATABASES.
15022	ZD020	Cannot find the user "{0}.{1}" in system table SYSUSERS.
15023	ZD021	"Cannot find the routine "{0}.{1}" with routine id {2} in system tables SYSROUTINES.
15024	ZD022	"Cannot find the trigger "{0}.{1}" in system table SYSTRIGGERS.
15025	ZD023	"Cannot find the routine "{0}" in system table SYSROUTINES using "{1}" as the existing CURRENT_PATH.
15026	ZD024	"The routine "{0}" already exists in system table SYSROUTINES using "{1}" as the existing CURRENT_PATH.
15027	ZD025	"The trigger "{0}" is already defined for table "{1}".

PointBase message code	***SQL state value***	***PointBase message***
15028	ZD026	"The external "{0}" routine had the following runtime exception "{1}"
15029	ZD027	"The external "{0}" routine could not be found at "{1}"
15030	ZD028	"The external "{0}" routine threw an exception: "{1}"
15031	ZD029	"The routine name "{0}" does not contain a "{1}" to delimit the method name.
15032	ZD030	"Invalid column "{0}" specified at position {1}. The column must be referenced in this specific join.
15033	ZD031	"Attempted to create a new database named "{0}" that is already opened.
15034	ZD032	"Attempted to create a new database named "{0}" with user: "{1}", password: "{2}". Only SYSADMIN can be used to create a database.
15035	ZD033	Incompatible data types in UNION select expressions.
15036	ZD034	The number of expressions in the SELECT statements in the UNION are different.
15037	ZD035	"Cannot find the schema with the id "{0}" in the system catalog.
15038	ZD036	"Cannot find the table in the schema "{0}" with the id "{1}}.
15039	ZD037	The Update statement Set clause found {0} target column(s) and {1} source expression(s). The same number of values must be found in each.
15040	ZD038	Only SysAdmin can drop the system table {0}.{1}.
15041	ZD039	Only SysAdmin can update the system table {0}.{1}.
15042	ZD040	The foreign key table {0}.{1} does not have a primary key match to the referencing table.
15043	ZD041	Invalid data type for a negate operation. The data type must be a type of the numeric family.
15044	ZD042	The Primary Key column {0} cannot have a default value of NULL.
15045	ZD043	Invalid Constraint Type of {0} was specified.
15046	ZD044	"Cannot find the routine "{0}" in system table SYSROUTINES using "{1}" as the existing CURRENT_PATH.

PointBase message code	SQL state value	PointBase message
15047	ZD045	"Invalid Foreign Key Constraint {0}. Number of columns in the parent and child are not equal. The parent has {1} columns and the child has {2} columns.
15048	ZD046	"Invalid Foreign Key Constraint {0}. The data type of the parent column {1} is not the same as of the child column {2}.
15049	ZD047	Access rule violation. User {0} is not the owner of the schema {1} and cannot create the SQL Object {2}.
15050	ZD048	Access rule violation. User {0} is not the owner of the schema {1} and cannot drop the SQL Object {2}.
15051	ZD049	Constraint definition column {0} is not a valid column of the table.
15052	ZD050	Column {0} occurs more than once in the same constraint definition.
15053	ZD051	The SQL Routine was defined to have an SQL body but the language specified was not SQL.
15054	ZD052	User {0} does not have the revoke privilege of {1} on object {2}. The privilege was never granted.
15055	ZD053	The columns in the USING clause do not constitute a valid join condition.
15056	ZD054	The constraint {0}.{1} was not found.
15057	ZD055	The constraint {0}.{1} already exists.
15058	ZD056	No Primary Key was found for table {0}.{1}. An attempt was made to find a primary key because the table had been used in the reference clause of a foreign key constraint.
15059	ZD057	The SQL Routine was defined to have a LANGUAGE of SQL but contains an external body.
15300	3F000	Invalid schema name. {0} specified at position {1}.
15302	0E000	Invalid schema name list {0}.
15303	ZD301	Access rule violation. User {0} is not the owner of the schema {1} and cannot create the SQL Object {2}.
20000	ZE000	query optimization errors. {0}.
25000	ZG000	Run-time execution errors. {0}.
25001	ZG001	An illegal attempt to perform an internal Bind command occurred.
25002	ZG002	An illegal attempt to perform an internal Describe command occurred.

PointBase message code	***SQL state value***	***PointBase message***
25003	ZG003	An illegal attempt to perform an internal Fetch command occurred.
25004	ZG004	Dynamic parameter markers and Bind variable count mismatch. Number of parameter markers: {0}. Number of bind variables: {1}
25005	ZG005	The following value: {0} is not a valid SQL Boolean value.
25006	ZG006	Not supported {0}
25007	ZG007	Illegal cast operation between type: {0} and type: {1}.
25008	ZG008	Attempted to insert a NULL value into a column that specified NOT NULL.
25009	ZG009	Attempt to turn off log in the middle of a transaction.
25010	ZG010	Only Select Statements are allowed in the "plan only" mode.
25011	ZG011	Subquery result is not a single scalar value.
25012	ZG012	Result Set {0} is an invalid/non-existent database result set.
25013	ZG013	An illegal attempt to perform an internal GetResultSet command occurred.
25015	ZG015	An attempt was made to convert between incompatible data types.
25021	ZG021	Invalid combination of isolation level and transaction access mode.
25101	3C000	Ambiguous cursor name. {0}.
25102	34000	Invalid cursor name. {0}.
25103	36001	Cursor sensitivity request rejected. {0}.
25104	36002	Cursor sensitivity request failed. {0}.
25105	24000	Invalid cursor state. {0}.
25201	22001	Data exception -- string data right truncation. {0}.
25202	22002	Data exception -- null value, no indicator parameter. {0}.
25203	22003	Data exception -- numeric value out of range. {0}.
25204	22004	Data exception -- null value not allowed. {0}.
25205	22005	Data exception -- error in assignment. {0}.

PointBase message code	***SQL state value***	***PointBase message***
25206	22007	Data exception -- invalid datetime format. {0}.
25207	22008	Data exception -- datetime field overflow. {0}.
25208	22009	Data exception -- invalid timezone displacement. {0}.
25209	22010	Data exception -- invalid indicator parameter value. {0}.
25210	22011	Data exception -- substring error. {0}.
25211	22012	Data exception -- division by zero. {0}.
25212	22014	Data exception -- invalid update value. {0}.
25213	22015	Data exception -- interval field overflow. {0}.
25214	22018	Data exception -- invalid character for cast. {0}.
25215	22019	Data exception -- invalid escape character. {0}.
25216	22020	Data exception -- invalid limit value. {0}.
25217	22022	Data exception -- indicator overflow. {0}.
25218	22023	Data exception -- invalid parameter value. {0}.
25219	22025	Data exception -- invalid escape sequence. {0}.
25220	22026	Data exception -- string data length mismatch. {0}.
25221	22027	Data exception -- trim error. {0}.
25222	22028	Data exception -- row already exists. {0}.
25223	2200A	Data exception -- null value in reference target. {0}.
25224	2201B	Data exception -- invalid regular expression. {0}.
25225	2201C	Data exception -- null row not permitted in table. {0}.
25226	2201D	Data Exception -- No Applicable Cast Operator. {0}
25227	2201E	User {0} already exists in the database.
25228	2201F	Schema {0} already exists in the database.
25229	22020	Duplicate row found in unique index.
25230	2201H	Trigger {0} already exists in the database.
25301	38001	External routine exception -- containing SQL not permitted. {0}.
25302	38002	External routine exception -- modifying SQL not permitted. {0}.
25303	38003	External routine exception -- prohibited SQL statement {0} not permitted.

PointBase message code	*SQL state value*	*PointBase message*
25304	38004	External routine exception -- reading SQL-data not permitted. {0}.
25305	39001	External routine invocation exception -- invalid SQLSTATE returned. {0}.
25306	39004	External routine invocation exception -- null value not allowed. {0}.
25307	2F002	SQL routine exception -- modifying SQL-data not permitted. {0}.
25308	2F003	SQL routine exception -- prohibited SQL-statement {0} attempted.
25309	2F004	SQL routine exception -- reading SQL-data not permitted. {0}.
25310	2F005	SQL routine exception -- function {0} executed no return statement.
25401	0B000	Invalid transaction initiation. {0}.
25402	25001	Invalid transaction state -- active SQL-transaction. {0}.
25403	25002	Invalid transaction state -- branch transaction already active. {0}.
25404	25003	Invalid transaction state -- inappropriate access mode for branch transaction. {0}.
25405	25004	Invalid transaction state -- inappropriate isolation level for branch transaction. {0}.
25406	25005	Invalid transaction state -- no active SQL-transaction for branch transaction. {0}.
25407	25006	Invalid transaction state -- read-only SQL-transaction. {0}.
25408	25007	Invalid transaction state -- schema and data statement mixing not supported. {0}.
25409	25008	Invalid transaction state -- held cursor requires same isolation level. {0}.
25410	2D000	Invalid transaction termination. {0}.
25411	40001	Transaction rollback serialization failure. {0}.
25412	40002	Transaction rollback integrity constraint violation. {0}.
25413	40003	Transaction rollback statement completion unknown. {0}.
25414	40004	Transaction rollback triggered action exception. {0}.

PointBase message code	***SQL state value***	***PointBase message***
25415	3B001	Savepoint invalid specification exception. {0}.
25416	3B002	Too many savepoints. {0}.
25417	3B003	Transaction Access Mode is READ_ONLY. No statement that modifies data is allowed in this mode.
25418	3B004	Data Log is off for this table. This statement is not allowed.
25501	09000	Triggered action exception. {0}.
25702	27000	Triggered data change violation. {0}.
26801	21000	Cardinality violation. {0}.
26802	44000	With check option violation. {0}.
26803	23000	Integrity constraint violation occurred with constraint {0}.
40000	ZL000	System catalog error. {0}.
40001	ZL001	System catalog internal error. {0}.
45000	ZM000	Space manager errors. {0}.
50000	ZN000	Cache Full. Current size is {0} pages. Increase the size of the cache using the cache.size=<number of pages>
50001	ZN001	Cache manager I/O error. {0}.
50002	ZN002	Cache manager did not find requested page. Either retry the system and try again or contact Technical Support.
50003	ZN003	Cache Manager Page Zero is a reserved page. Either retry the system and try again or contact Technical Support.
50004	ZN004	Cache Manager Commit detected not all pages have zero ref count first offending page number {0}. Either retry the system and try again or contact Technical Support.
50005	ZN005	Page factory is null. Either retry the system and try again or contact Technical Support.
50006	ZN006	Database {0} does not exist or cannot be found in home {1} or either specify database.home=<folder> in `pointbase.ini` to indicate the database folder.
50008	ZN008	An attempt (on {0}) to create a database failed because the database already exists.
50009	ZN009	Page has a negative ref count: {0} {1}. Either retry the system and try again or contact Technical Support.
50010	ZN010	No cache context has been set. Either retry the system and try again or contact Technical Support.

PointBase message code	SQL state value	PointBase message
50011	ZN011	Database exceeds maximum size. This version {0} supports a maximum database size of {1} MB.
50012	ZN012	SEVERE ERROR!!! The database file [{0}] has failed an integrity check. A possible reason for this is a change in page size. See the documentation for control over this parameter. The database will be shutdown.
50013	ZN013	The database lock file [{0}] indicates that the database is currently open in another process.
55000	ZP000	Transaction manager errors. {0}.
60000	ZQ000	Log manager errors. {0}.
65000	ZR000	Recovery manager errors. {0}.
70000	ZS000	Replication manager errors. {0}.
70001	ZS001	Filtering table {0} not specified in the list of table referenced. One must specify tables in the list of the command.
70002	ZS002	The table {0} specified in the UNISYNC command cannot be found. Be sure that the table exists and the user has synchronization privileges.
75000	ZT000	File I/O errors. {0}.
75001	ZT001	Character encoding {0} is not supported by the system.
76000	ZU000	The properties file ({0}) encountered a problem: {1}. The `pointbase.ini` file has been specified incorrectly.
76001	ZU001	Class creation ({0}) error: {1}. Either retry the system and try again or contact Technical Support.
77000	ZU000	The sort could not complete because memory was exceeded. Try decreasing sort.size=<KB> in `pointbase.ini`
77001	ZU001	No such element. (nextElement cannot go beyond end of collection)
77002	ZU002	The elements could not be cast to the appropriate sort interface
77003	ZU003	I/O Exception occurred during external sorting {0}
78000	ZW000	Operation {0} not permitted on incompatible types {1} {2}
78001	ZW000	Marker type cannot be determined. The data type of the dynamic parameter marker is an invalid SQL data type.
78002	ZW000	The data type of the operand is an invalid SQL data type.

PointBase message code	***SQL state value***	***PointBase message***
78003	ZW003	The value "{0}" cannot be converted to a number.
78004	ZW004	The default value "{0}" cannot be converted to {1}.
78005	ZW005	The expression "{0}" cannot be used in a comparison predicate with "{1}" because they have incompatible collation sequences.
79000	ZY000	Startup Exception: {0}. Either retry the system and try again or contact Technical Support.
79001	ZY001	Database Initialization failed. Reason: {0}.
79002	ZY002	Database {0} already exists. Drop the database and recreate it.
81000	ZZ000	Failed to insert key into btree page. This is a fatal error.
82000	ZZB00	Specified LOB offset {0} is out of range. Valid values are from 0 to {1}.
82001	ZZB01	Failed to read LOB data from input stream. IOException = {0}.
82002	ZZB02	Data Exception -- data is a stream of bytes for parameter {0}
82003	ZZB03	Attempted to convert a large lob to an in-memory string. Maximum size for conversion is {0}.
83000	ZZC00	Method called on an object not found in the object map.
83001	ZZC01	No method of this name found on this object
83002	ZZC02	An IO Exception occurred: {0}
83003	ZZC03	The maximum number of connections {0} has been exceeded.
84000	ZZC01	Lock time out; try later.
84001	ZZC02	Lock Promotion timed out; try later.
84002	ZZC03	Lock escalation failed; try later.
84003	ZZC04	LockManager Internal error; Table lock not found.
84004	ZZC05	LockManager Internal error; table lock incompatible.
85000	ZZE00	Referential Integrity Violation. {0}.{1} references {2}.{3}
86000	ZZF01	IO Exception when creating a Blob object : {0}
86001	ZZF02	Requested length on {0} of a Blob is too large.
86002	ZZF03	IO Exception when creating a Clob object: {0}

PointBase message code	SQL state value	PointBase message
86003	ZZF04	IO Exception occurred on {0} of a Clob: {1}
86004	ZZF05	Requested length on {0} of a Clob is too large.
86005	ZZF06	This statement is closed.
86006	ZZF07	Select Statement is not allowed in executeUpdate or executeBatch.
86007	ZZF08	Set Auto commit to false before you perform executeBatch.
86008	ZZF09	There are no Statements available to perform executeBatch.
86009	ZZF10	{0} not supported.
86010	ZZF11	JDBC {0} Core API Method not yet supported.
86011	ZZF12	There are no bind variables to bind with the statement.
86012	ZZF13	Object {0} not serializable.
86013	ZZF14	IOException occurred on {0} of PreparedStatement : {1}
86014	ZZF15	Parameter Index {0} exceeds the number of bind variables
86015	ZZF16	Column [{0}] does not exists in the result set.
86016	ZZF17	Date format error : {0}
86017	ZZF18	Number Format error: {0}
86018	ZZF19	This result set has been invalidated.
86019	ZZF20	IO Exception occurred on {0} of a ResultSet: {1}
86020	ZZF21	Syntax error in parsing native SQL statement within : {0}
86021	ZZF22	IOException occurred on {0} of ResultSet : {1}
86022	ZZF23	IOException occurred on {0} of JDBCPrimitives: {1}
86023	ZZF24	Malformed URL
86024	ZZF25	{0} server rejected SQL connection.
86025	ZZF26	{0} server returned unexpected data
86026	ZZF27	{0} server is a newer version than {0} client
86027	ZZF28	IO Exception when creating a netJDBCConnection object: {0}
86028	ZZF29	Result set is in an invalid state. May be before the first row or after the last row.
86029	ZZF30	{0}(String) method not allowed on a Prepared Statement.

PointBase message code	***SQL state value***	***PointBase message***
86030	ZZF31	Execution cancelled due to autocommit or user request.
86031	ZZF32	Fetch size must be greater or equal to 0
86032	ZZF33	Max rows must be greater or equal to 0
86033	ZZF33	Field size must be greater or equal to 0
86034	ZZF34	Only Select Statements are allowed in the executeQuery.
86035	ZZF35	Query time out value should be greater or equal to 0
86036	ZZF36	The referencing alias {0} is not supported.
86037	ZZF37	{0} assignment allowed for before triggers only.
86038	ZD031	"Attempted to open a database named "{0}" when another database named "{1}" is already open.
86039	ZZF38	Wrong pattern [{0}] provided in [{1}] for [{2}].
86040	ZZF39	Invalid Parameter [{0}] in [{1}] for [{2}]. [{3}] not allowed.
86041	ZZF40	Invalid Array parameter [{0}] in [{1}] for [{2}]. Length should be greater than 0.
90000	ZZF90	shutdown called on the server.
90001	ZZF91	shutdown force called on the server.
90002	ZZF92	Can not shutdown the server. {0} users currently connected. Use shutdown force to shutdown.
90003	ZZF93	You are not allowed to create a new database.
90004	ZZF94	Maximum supported database size has been reached.
90005	ZZF95	Can not shutdown the database. {0} connections currently opened. Use shutdown force to shutdown database.
90006	ZZF96	Database Server is shutdown

Appendix C: Country and Language Codes

This section lists valid Language Codes and valid Country Codes.

Language Codes

(AFAN) OROMO	OM
ABKHAZIAN	AB
AFAR	AA
AFRIKAANS	AF
ALBANIAN	SQ
AMHARIC	AM
ARABIC	AR
ARMENIAN	HY
ASSAMESE	AS
AYMARA	AY
AZERBAIJANI	AZ
BASHKIR	BA
BASQUE	EU
BENGALI; BANGLA	BN
BHUTANI	DZ
BIHARI	BH
BISLAMA	BI
BRETON	BR
BULGARIAN	BG

BURMESE	MY
BYELORUSSIAN	BE
CAMBODIAN	KM
CATALAN	CA
CHINESE	ZH
CORSICAN	CO
CROATIAN	HR
CZECH	CS
DANISH	DA
DUTCH	NL
ENGLISH	EN
ESPERANTO	EO
ESTONIAN	ET
FAROESE	FO
FIJI	FJ
FINNISH	FI
FRENCH	FR
FRISIAN	FY
GALICIAN	GL
GEORGIAN	KA
GERMAN	DE
GREEK	EL
GREENLANDIC	KL
GUARANI	GN
GUJARATI	GU
HAUSA	HA
HEBREW (FORMERLY IW)	HE
HINDI	HI
HUNGARIAN	HU
ICELANDIC	IS

INDONESIAN (FORMERLY IN)	ID
INTERLINGUA	IA
INTERLINGUE	IE
INUKTITUT	IU
INUPIAK	IK
IRISH	GA
ITALIAN	IT
JAPANESE	JA
JAVANESE	JW
KANNADA	KN
KASHMIRI	KS
KAZAKH	KK
KINYARWANDA	RW
KIRGHIZ	KY
KIRUNDI	RN
KOREAN	KO
KURDISH	KU
LAOTHIAN	LO
LATIN	LA
LATVIAN, LETTISH	LV
LINGALA	LN
LITHUANIAN	LT
MACEDONIAN	MK
MALAGASY	MG
MALAY	MS
MALAYALAM	ML
MALTESE	MT
MAORI	MI
MARATHI	MR
MOLDAVIAN	MO

MONGOLIAN	MN
NAURU	NA
NEPALI	NE
NORWEGIAN	NO
OCCITAN	OC
ORIYA	OR
PASHTO, PUSHTO	PS
PERSIAN	FA
POLISH	PL
PORTUGUESE	PT
PUNJABI	PA
QUECHUA	QU
RHAETO-ROMANCE	RM
ROMANIAN	RO
RUSSIAN	RU
SAMOAN	SM
SANGHO	SG
SANSKRIT	SA
SCOTS GAELIC	GD
SERBIAN	SR
SERBO-CROATIAN	SH
SESOTHO	ST
SETSWANA	TN
SHONA	SN
SINDHI	SD
SINHALESE	SI
SISWATI	SS
SLOVAK	SK
SLOVENIAN	SL
SOMALI	SO

SPANISH	ES
SUNDANESE	SU
SWAHILI	SW
SWEDISH	SV
TAGALOG	TL
TAJIK	TG
TAMIL	TA
TATAR	TT
TELUGU	TE
THAI	TH
TIBETAN	BO
TIGRINYA	TI
TONGA	TO
TSONGA	TS
TURKISH	TR
TURKMEN	TK
TWI	TW
UIGHUR	UG
UKRAINIAN	UK
URDU	UR
UZBEK	UZ
VIETNAMESE	VI
VOLAPUK	VO
WELSH	CY
WOLOF	WO
XHOSA	XH
YIDDISH (FORMERLY JI)	YI
YORUBA	YO
ZHUANG	ZA
ZULU	ZU

Country Codes

The following table lists valid country codes.

AFGHANISTAN	AF
ALBANIA	AL
ALGERIA	DZ
AMERICAN SAMOA	AS
ANDORRA	AD
ANGOLA	AO
ANGUILLA	AI
ANTARCTICA	AQ
ANTIGUA AND BARBUDA	AG
ARGENTINA	AR
ARMENIA	AM
ARUBA	AW
AUSTRALIA	AU
AUSTRIA	AT
AZERBAIJAN	AZ
BAHAMAS	BS
BAHRAIN	BH
BANGLADESH	BD
BARBADOS	BB
BELARUS	BY
BELGIUM	BE
BELIZE	BZ
BENIN	BJ
BERMUDA	BM
BHUTAN	BT
BOLIVIA	BO
BOSNIA AND HERZEGOWINA	BA

BOTSWANA	BW
BOUVET ISLAND	BV
BRAZIL	BR
BRITISH INDIAN OCEAN TERRITORY	IO
BRUNEI DARUSSALAM	BN
BULGARIA	BG
BURKINA FASO	BF
BURUNDI	BI
CAMBODIA	KH
CAMEROON	CM
CANADA	CA
CAPE VERDE	CV
CAYMAN ISLANDS	KY
CENTRAL AFRICAN REPUBLIC	CF
CHAD	TD
CHILE	CL
CHINA	CN
CHRISTMAS ISLAND	CX
COCOS (KEELING) ISLANDS	CC
COLOMBIA	CO
COMOROS	KM
CONGO	CG
COOK ISLANDS	CK
COSTA RICA	CR
COTE D'IVOIRE	CI
CROATIA (LOCAL NAME: HRVATSKA)	HR
CUBA	CU
CYPRUS	CY
CZECH REPUBLIC	CZ
DENMARK	DK

DJIBOUTI	DJ
DOMINICA	DM
DOMINICAN REPUBLIC	DO
EAST TIMOR	TP
ECUADOR	EC
EGYPT	EG
EL SALVADOR	SV
EQUATORIAL GUINEA	GQ
ERITREA	ER
ESTONIA	EE
ETHIOPIA	ET
FALKLAND ISLANDS (MALVINAS)	FK
FAROE ISLANDS	FO
FIJI	FJ
FINLAND	FI
FRANCE	FR
FRANCE, METROPOLITAN	FX
FRENCH GUIANA	GF
FRENCH POLYNESIA	PF
FRENCH SOUTHERN TERRITORIES	TF
GABON	GA
GAMBIA	GM
GEORGIA	GE
GERMANY	DE
GHANA	GH
GIBRALTAR	GI
GREECE	GR
GREENLAND	GL
GRENADA	GD
GUADELOUPE	GP

GUAM	GU
GUATEMALA	GT
GUINEA	GN
GUINEA-BISSAU	GW
GUYANA	GY
HAITI	HT
HEARD AND MC DONALD ISLANDS	HM
HONDURAS	HN
HONG KONG	HK
HUNGARY	HU
ICELAND	IS
INDIA	IN
INDONESIA	ID
IRAN (ISLAMIC REPUBLIC OF)	IR
IRAQ	IQ
IRELAND	IE
ISRAEL	IL
ITALY	IT
JAMAICA	JM
JAPAN	JP
JORDAN	JO
KAZAKHSTAN	KZ
KENYA	KE
KIRIBATI	KI
KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF	KP
KOREA, REPUBLIC OF	KR
KUWAIT	KW
KYRGYZSTAN	KG
LAO PEOPLE'S DEMOCRATIC REPUBLIC	LA
LATVIA	LV

LEBANON	LB
LESOTHO	LS
LIBERIA	LR
LIBYAN ARAB JAMAHIRIYA	LY
LIECHTENSTEIN	LI
LITHUANIA	LT
LUXEMBOURG	LU
MACAU	MO
MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	MK
MADAGASCAR	MG
MALAWI	MW
MALAYSIA	MY
MALDIVES	MV
MALI	ML
MALTA	MT
MARSHALL ISLANDS	MH
MARTINIQUE	MQ
MAURITANIA	MR
MAURITIUS	MU
MAYOTTE	YT
MEXICO	MX
MICRONESIA, FEDERATED STATES OF	FM
MOLDOVA, REPUBLIC OF	MD
MONACO	MC
MONGOLIA	MN
MONTSERRAT	MS
MOROCCO	MA
MOZAMBIQUE	MZ
MYANMAR	MM
NAMIBIA	NA

NAURU	NR
NEPAL	NP
NETHERLANDS	NL
NETHERLANDS ANTILLES	AN
NEW CALEDONIA	NC
NEW ZEALAND	NZ
NICARAGUA	NI
NIGER	NE
NIGERIA	NG
NIUE	NU
NORFOLK ISLAND	NF
NORTHERN MARIANA ISLANDS	MP
NORWAY	NO
OMAN	OM
PAKISTAN	PK
PALAU	PW
PANAMA	PA
PAPUA NEW GUINEA	PG
PARAGUAY	PY
PERU	PE
PHILIPPINES	PH
PITCAIRN	PN
POLAND	PL
PORTUGAL	PT
PUERTO RICO	PR
QATAR	QA
REUNION	RE
ROMANIA	RO
RUSSIAN FEDERATION	RU
RWANDA	RW
SAINT KITTS AND NEVIS	KN

SAINT LUCIA	LC
SAINT VINCENT AND THE GRENADINES	VC
SAMOA	WS
SAN MARINO	SM
SAO TOME AND PRINCIPE	ST
SAUDI ARABIA	SA
SENEGAL	SN
SEYCHELLES	SC
SIERRA LEONE	SL
SINGAPORE	SG
SLOVAKIA (SLOVAK REPUBLIC)	SK
SLOVENIA	SI
SOLOMON ISLANDS	SB
SOMALIA	SO
SOUTH AFRICA	ZA
SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS	GS
SPAIN	ES
SRI LANKA	LK
ST. HELENA	SH
ST. PIERRE AND MIQUELON	PM
SUDAN	SD
SURINAME	SR
SVALBARD AND JAN MAYEN ISLANDS	SJ
SWAZILAND	SZ
SWEDEN	SE
SWITZERLAND	CH
SYRIAN ARAB REPUBLIC	SY
TAIWAN, PROVINCE OF CHINA	TW
TAJIKISTAN	TJ

TANZANIA, UNITED REPUBLIC OF	TZ
THAILAND	TH
TOGO	TG
TOKELAU	TK
TONGA	TO
TRINIDAD AND TOBAGO	TT
TUNISIA	TN
TURKEY	TR
TURKMENISTAN	TM
TURKS AND CAICOS ISLANDS	TC
TUVALU	TV
UGANDA	UG
UKRAINE	UA
UNITED ARAB EMIRATES	AE
UNITED KINGDOM	GB
UNITED STATES	US
UNITED STATES MINOR OUTLYING ISLANDS	UM
URUGUAY	UY
UZBEKISTAN	UZ
VANUATU	VU
VATICAN CITY STATE (HOLY SEE)	VA
VENEZUELA	VE
VIET NAM	VN
VIRGIN ISLANDS (BRITISH)	VG
VIRGIN ISLANDS (U.S.)	VI
WALLIS AND FUTUNA ISLANDS	WF
WESTERN SAHARA	EH
YEMEN	YE
YUGOSLAVIA	YU

ZAIRE	ZR
ZAMBIA	ZM
ZIMBABWE	ZW