

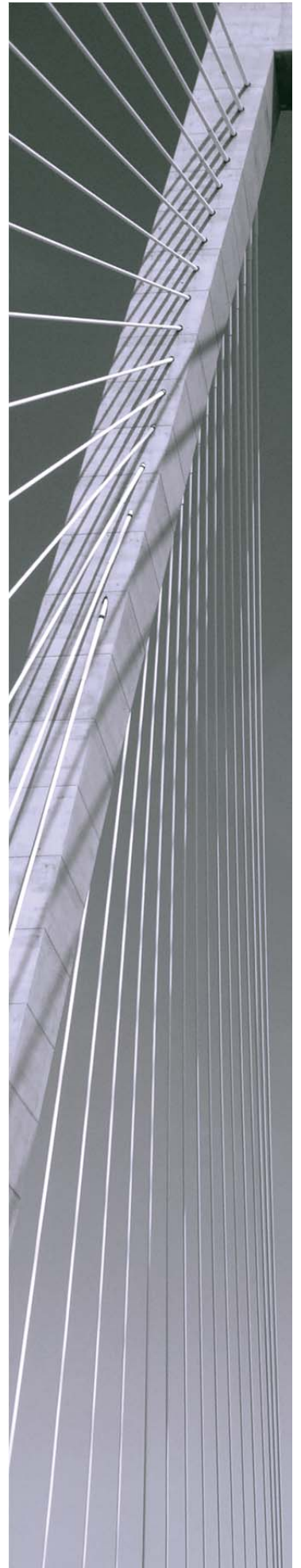


# **SimbaEngine SDK 8**

## **SimbaClientServer User's Guide**

**For version 8.1.0**

**Simba Technologies Inc.**



Copyright ©2009–2011 Simba Technologies Inc. All Rights Reserved.

Information in this document is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. No part of this publication, or the software it describes, may be reproduced, transmitted, transcribed, stored in a retrieval system, decompiled, disassembled, reverse-engineered, or translated into any language in any form by any means for any purpose without the express written permission of Simba Technologies Inc.

### Simba Trademarks

Simba, the Simba logo, SimbaEngine, SimbaEngine C/S, SimbaClient, SimbaD20, SimbaEngine SDK and SimbaODBC are registered trademarks of Simba Technologies Inc. All other trademarks and/or servicemarks are the property of their respective owners.

### Simba Technologies Inc.

938 West 8<sup>th</sup> Avenue  
Vancouver, BC Canada  
V5Z 1E5

Tel. +1.604.633.0008  
Fax. +1.604.633.0004

[www.simba.com](http://www.simba.com)

Printed in Canada

## Third Party Trademarks

Copyright (c) 1995-2010 International Business Machines Corporation and others

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment:  
"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [openssl-core@openssl.org](mailto:openssl-core@openssl.org).
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment:  
"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

## Table of Contents

- 1 Before You Begin..... 2
  - 1.1 Who Should Read this Guide ..... 2
- 2 Introducing SimbaServer..... 3
  - 2.1 The Modern Server-based DBMS..... 3
  - 2.2 Client/Server Remote Protocols ..... 4
- 3 Working with SimbaServer ..... 6
  - 3.1 Development Strategy ..... 7
  - 3.2 Testing Your Server..... 8
- 4 Installing SimbaServer ..... 9
  - 4.1 Preparation ..... 9
  - 4.2 Installation Procedure..... 12
- 5 Configuring SimbaServer..... 15
  - 5.1 SimbaServer on Windows..... 15
  - 5.2 SimbaServer on UNIX and Linux ..... 16
  - 5.3 SimbaServer Configuration Options ..... 16
- 6 Configuring SimbaClient..... 23
  - 6.1 SimbaClient for ODBC ..... 23
  - 6.2 SimbaClient for JDBC ..... 30
- 7 Configuring Secure Sockets Layer (SSL) ..... 36
  - 7.1 Generating a Certificate Authority (CA) Certificate –for self-signing ..... 36
  - 7.2 Generating an SSL Certificate with Verisign..... 37
  - 7.3 Creating a Trusted Key Store for JDBCClient..... 39
  - 7.4 Distributing SSL Certificates ..... 39
- 8 Enabling Logging ..... 39
  - 8.1 SimbaServer ..... 40
  - 8.2 SimbaClient for ODBC ..... 41
  - 8.3 SimbaClient for JDBC ..... 41
- 9 Advanced Information..... 41
  - 9.1 SimbaServer Architecture ..... 42
  - 9.2 SimbaClient for ODBC Architecture..... 43
- Appendix A: How Do I Get Support?..... 45

## Table of Figures

- Figure 1: A modern, server-based RDBMS with access via a remote protocol. .... 3
- Figure 2: The relationship between the SimbaServer component deployment and the Simba Message Protocol layers. .... 4
- Figure 3: The Simba protocol layers showing how they nest together like letters inside of envelopes. .... 5
- Figure 4: A comparison of the stand-alone Simba SQL Engine ODBC driver and an equivalent client/server solution. The components of the stacks are lined up by their functionality. .... 6

# 1 Before You Begin

With SimbaClient/Server, you can quickly and efficiently provide data access to connect ODBC applications to remote data stores in networked environments.

SimbaClient/Server includes networking components that allow direct communication using TCP/IP between client computers and a Simba SQL Engine and Data Store Interface (DSI) implementation on the server.

## 1.1 Who Should Read this Guide

We have addressed this SimbaServer User's Guide to our customers who have completed building and testing a prototype ODBC driver and who are ready to test it in a client/server environment. We have also addressed it to those who want to explore how SimbaClient/Server will work in their environment, using one of the Quickstart example DSI implementations. If you want more information on building and testing a prototype ODBC driver, refer to the Getting Started Guide. If you want to get started right away learning how to create an example SimbaServer to experiment with, please read Section 3, Working with SimbaServer, below.

The following chapters in this User's Guide contain information explaining the creation, installation, configuration, and administration of the server and client components of the SimbaEngine SDK.

This User's Guide assumes that you have a working understanding of modern database principles and terminology, the C++ programming language, and your development environment.

## 2 Introducing SimbaServer

SimbaServer turns your Data Store Interface (DSI) implementation into a full-featured, remote RDBMS. The same DSI implementation you use to create an ODBC 3.52 stand-alone driver can be used with SimbaServer to create a database server you can deploy on any platform supported by SimbaEngine SDK. Your new database server can be accessed by SimbaClients for ODBC and JDBC from any platform supported by the SimbaEngine SDK.

Server solutions created with SimbaServer have the same simple structure as the stand-alone ODBC 3.52 drivers you create. Being able to use the same DSI code for building a server that you develop as a stand-alone driver means that it is easier to debug and fix your DSI code and deliver a correct and robust product to your customer. The SimbaClient for ODBC uses the same SimbaODBC component as used in your stand-alone ODBC driver, so your customer will see the same behavior whether they use the stand-alone driver or a client/server solution.

### 2.1 The Modern Server-based DBMS

Modern Relational Database Management Systems (RDBMSs) provide access for users via a remote network protocol that runs on common networks such as TCP/IP. This provides nearly universal access to the RDBMS since a well-designed database protocol will run on most networks, and virtually all user machines are already networked. All major commercial RDBMSs work this way.

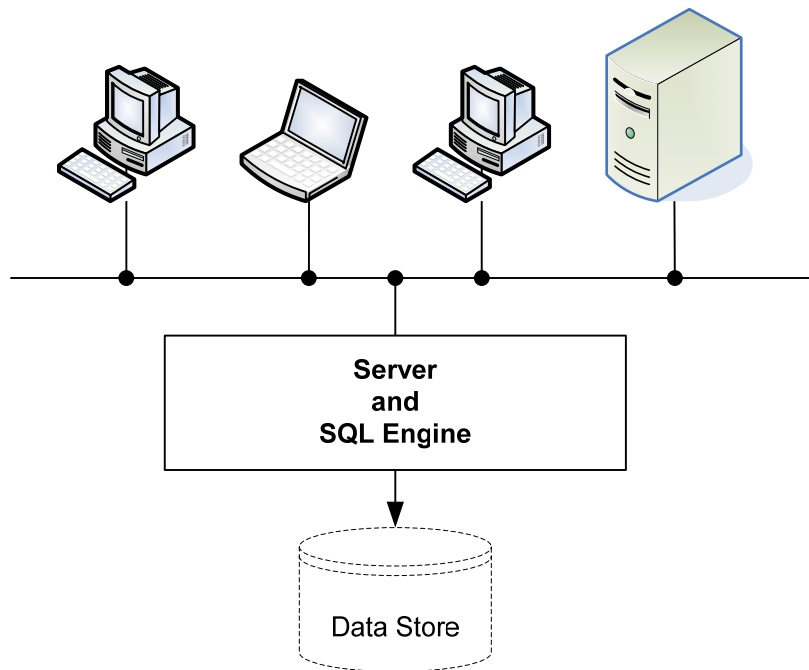


Figure 1: A modern, server-based RDBMS with access via a remote protocol.

Database access via a remote protocol also introduces tremendous flexibility in the choice of deployment architectures. This is because a remote network protocol creates an interface that is independent of language, operating system, word length, processor, and network. A well-designed remote protocol will allow any two machines to communicate. The result is that your clients and servers can be deployed where they make the most sense to your customers. Your customers can deploy servers on high-powered, highly reliable machines, and deploy clients where the users are.

SimbaServer turns your data store into a complete, server-based RDBMS that can be accessed by client machines from anywhere on your customer's network. Figure 1, above, illustrates the flexibility of deployment with a server-based RDBMS.

## 2.2 Client/Server Remote Protocols

The multiple advantages of a networked RDBMS make the remote protocol very important. The remote protocol must be able to run on any common network protocol while remaining independent of it. It must also be carefully designed with the requirements of data access in mind, which are quite different from the requirements of other remote protocols such as HTTP or RPC. Except in special circumstances, the protocol needs to be able to retrieve large amounts of data from the server without unacceptable delay and without hogging the network. The protocol also needs to be able to maintain a stateful connection between the client and server without exchanging too many messages.

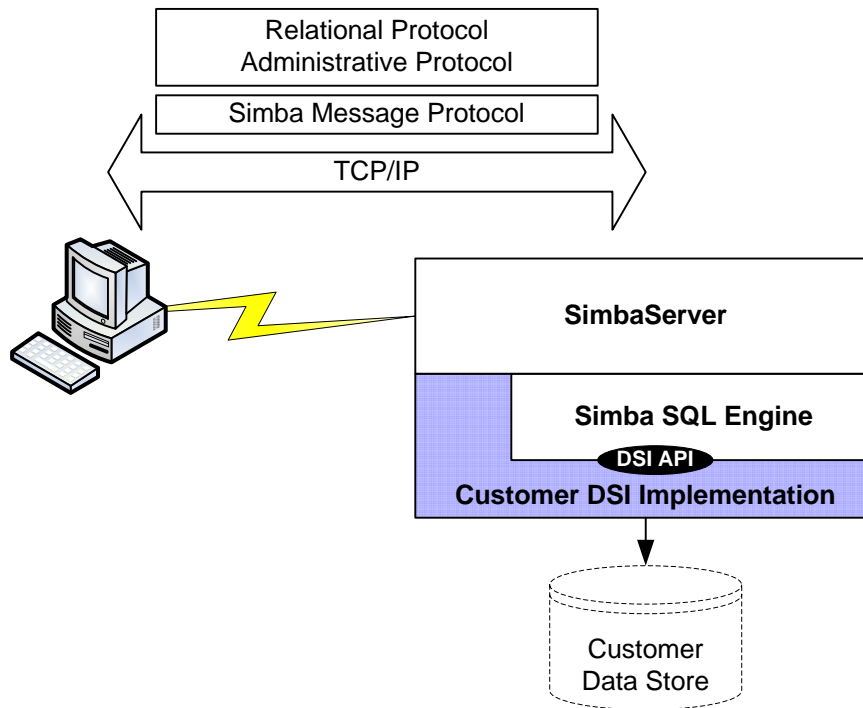


Figure 2: The relationship between the SimbaServer component deployment and the Simba Message Protocol layers.

SimbaServer uses a sophisticated and high-performance proprietary remote protocol to connect clients to servers. We have designed it specifically for database access and control. You can retrieve large volumes of data from the server in a short time while the operations that set up queries and maintain state are small, efficient and infrequent. The carefully designed layered approach makes this protocol flexible and robust. Figure 2, above, shows the relationship of the Simba protocols to the deployment, and the layered nature of the Simba protocol.

The Simba protocol consists of a basic protocol that runs on the local network, and more specialized protocols that run on top of that. In this way the specialized protocols that perform the useful work are isolated from any changes in the underlying network. Also, other specialized protocols can be added to the system without disturbing existing protocols, which allows for easy and reliable upgrades.

The Simba protocols work like other layered protocols in that they encapsulate the more specialized protocols inside the more general protocols. This is often compared to messages inside envelopes. You can see this in Figure 3, below.

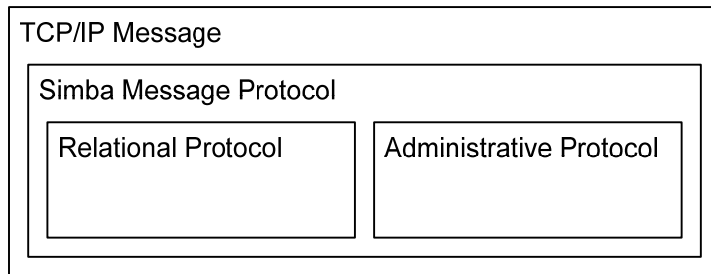


Figure 3: The Simba protocol layers showing how they nest together like letters inside of envelopes.

### 3 Working with SimbaServer

The architecture of a complete SimbaServer solution is very similar to that of a stand-alone ODBC driver. All of the same functionality is present with the addition of the client/server functionality that transports the DSI functionality across the network. From an internal point of view, SimbaServer is simpler than a stand-alone driver is because it does not include SimbaODBC. However, from the point of view of the developer they are the same.

This section describes how to link a DSII to create a SimbaServer executable and points to examples. It also describes how to test your new SimbaServer executable, provides advice on how to approach migrating your DSII from a stand-alone driver to SimbaServer, and gives some tips on making sure your DSII will work well with SimbaServer.

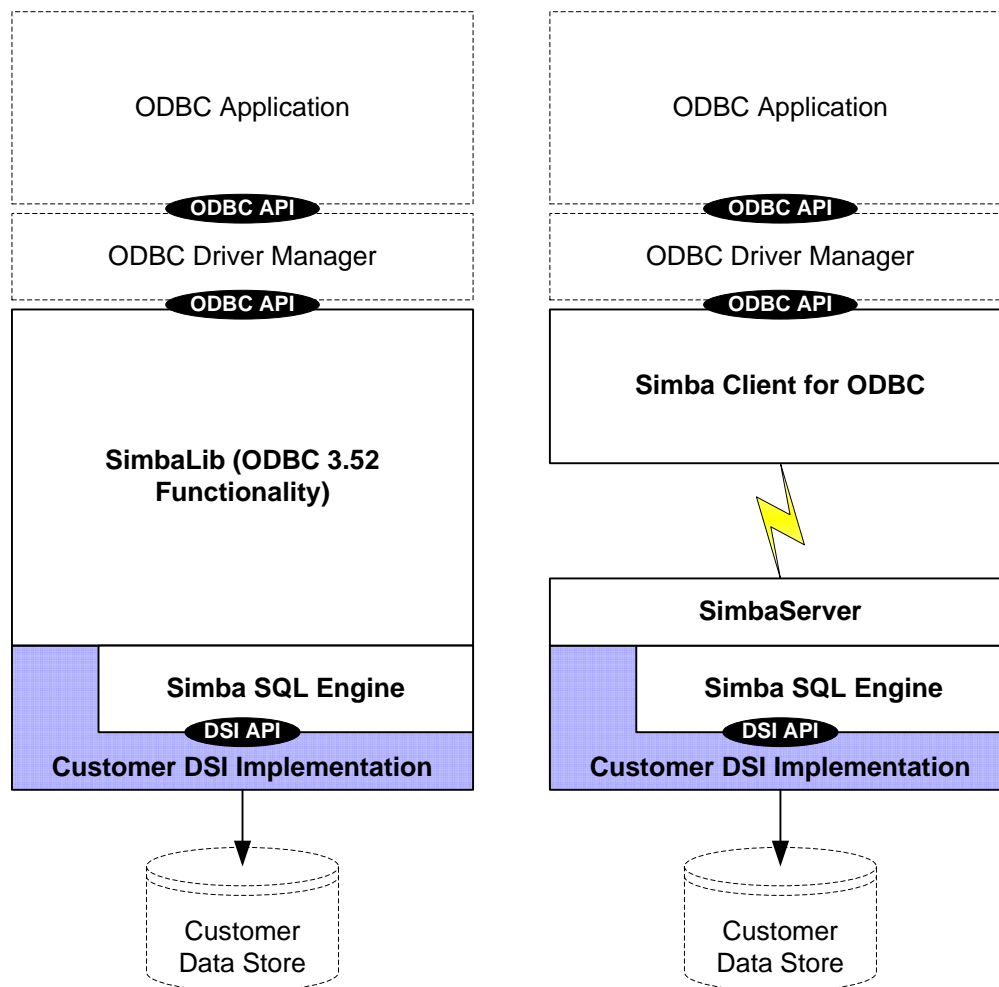


Figure 4: A comparison of the stand-alone Simba SQL Engine ODBC driver and an equivalent client/server solution. The components of the stacks are lined up by their functionality.

Figure 4, above, compares the stand-alone ODBC driver stack with the client/server stack to show the similarities. The same data access tools are used in the same roles in the two stacks

so you can be certain that if your stand-alone driver is correct, a client/server version will be correct. In Figure 4, the ODBC application and the driver manager are the same. Note that there is only one driver manager in the client/server stack. Some ODBC client/server architectures have two driver managers in the stack which introduces the problem of keeping them synchronized. The top end of the SimbaClient for ODBC uses the same SimbaODBC tool as the top end of the stand-alone driver. This means that their response to ODBC function calls will be the same.

The section of code that connects the ODBC functionality to the DSI API is the only place where the two stacks differ. In the stand-alone driver, some simple code connects the SimbaODBC code to the DSI API. However, in the client/server stack the entire client/server connection mechanism exists in that space. This is the ODBC client and the server tools that talk to each other and effectively project the DSI API across a network.

To create a server from your DSI implementation you will link your DSI code with the following libraries:

- SimbaServer.lib
- SimbaCommunications.lib
- SimbaMessages.lib

The project files and make files included in SimbaEngine SDK are set up to perform this step. All you need is your working DSI implementation.

The result is an executable (\*.exe on Windows). On Windows, Linux and UNIX you can run this executable from a user's command line. This makes it easy to start and stop the executable for testing. On Windows, you can also run the server executable as a service. The usual Windows service entry points are built into the command line executable.

## 3.1 Development Strategy

Because the component stacks of the stand-alone ODBC driver and the client/server are so similar, it makes sense to reduce the effort of creating a client/server solution by reducing the complexity of the system as you are building it. This means building a stand-alone ODBC driver and testing it for correctness and reliability before rebuilding your DSI implementation into a server and testing that system.

The Microsoft Visual Studio project files in the SimbaEngine Codebase, Quickstart and other example drivers have targets to build both stand-alone driver and servers using the same DSI implementation. For instance, you can follow the step-by-step plan for prototyping a driver in five days and make sure that it is working correctly when you are done. Then you can change the target build to Debug\_Server or Release\_Server and rebuild your DSI implementation into a

SimbaServer executable. The resulting server executable should have exactly the same responses to queries and the data store as does the stand-alone driver.

One way to reduce the effort of learning how to configure and troubleshoot the SimbaServer component is to start with the SimbaEngine Quickstart Driver example. This ODBC driver uses local files as database tables so it does not have to connect to a remote data store. In addition, the DSI Quickstart implementation is tested and guaranteed to work, so all you need to focus on is configuring the server and making sure the client can find it.

Begin your exploration of SimbaClient/Server by building the Server version of the SimbaEngine Quickstart Driver example and getting that working. When you can connect to the server from the client and retrieve data, you know that the system is working. This can act as a known-good system and you can troubleshoot the version of SimbaServer that works with your data store by comparing it to the known good system. In this way, you can take smaller steps and make more steady progress towards a working SimbaServer and a reliable client server system.

## 3.2 Testing Your Server

Because the stand-alone ODBC driver and SimbaServer with the same DSI implementation are so similar, you can use both of them tactically to reduce your testing complexity. Test the stand-alone driver first, and debug and fix the problems you find there before testing the SimbaServer version. This ensures that your DSI implementation is correct and reliable before you introduce the complexity of the client/server components.

Another consideration is that a stand-alone ODBC driver built with your DSI implementation should deliver exactly the same results as the SimbaServer version. Any differences points to an underlying problem that you should investigate.

## 4 Installing SimbaServer

This section describes how to install SimbaServer and the SimbaClients at your customer's site. For installation on Windows, this most often means writing an installer for the server executable and making sure that the entire configuration is properly performed. However, for Linux and UNIX it may mean shipping a tar.gz file and a list of instructions. In either case, this section will help you.

The rest of this section assumes that you want to create an installer for your server, or to create an installation package that your customers can use easily.

### 4.1 Preparation

The first step in setting up for installation, whether creating an automatic installer or simply a tar.gz file, is to list and gather all the files that you need for the installation. In the case of SimbaServer, this includes your new executable, SimbaClients for ODBC and JDBC, ICU libraries, SSL libraries, error message files and localization files. This subsection enumerates what you will need for the server and for the clients.

Note that some of the files for the server installation and the SimbaClient for ODBC will be the same, particularly the ICU translation libraries, the error messages and possibly the localization files. Just be aware that you may have to make sure that the same files are going into two different installers.

If you are using a .NET 2008 version of SimbaEngine SDK, you will require the Microsoft C++ 2008 Redistributable Package installed on any machine where SimbaClient for ODBC or SimbaServer will run. You can download this package for 32-bit and 64-bit Windows, free, from Microsoft. Here are the links:

vcredist\_x86.exe (32-bit):

<http://www.microsoft.com/DOWNLOADS/details.aspx?FamilyID=9b2da534-3e03-4391-8a4d-074b9f2bc1bf&displaylang=en>

vcredist\_x64.exe (64-bit):

<http://www.microsoft.com/DOWNLOADS/details.aspx?familyid=BD2A6171-E2D6-4230-B809-9A8D7548C1B6&displaylang=en>

If you are using a .NET 2005 version of SimbaEngine SDK, you will require the Microsoft C++ 2008 Redistributable Package installed on any machine where SimbaClient for ODBC or SimbaServer will run. You can download this package for 32-bit and 64-bit Windows, free, from Microsoft. Here are the links:

vcredist\_x86.exe (32-bit):

<http://www.microsoft.com/downloads/details.aspx?familyid=32bc1bee-a3f9-4c13-9c99-220b62a191ee&displaylang=en>

vcredist\_x64.exe (64-bit):

<http://www.microsoft.com/downloads/details.aspx?familyid=90548130-4468-4BBC-9673-D6ACABD5D13B&displaylang=en>

## SimbaServer Required Files

You will need the following files on Windows to ensure that SimbaServer will work properly when installed at your customer's site:

Windows File	Description
<Your DSI>SimbaServer.exe	Your new server executable with your DSI implementation. For example, the Quickstart executable is named QuickstartDSIServer.exe.
simbaicudt38_32.dll (simbaicudt38_64.dll if using 64-bit)	ICU utility libraries
simbaicu38_32.dll (simbaicu38_64.dll if using 64-bit)	
simbaicuuc38_32.dll (simbaicu38_64.dll if using 64-bit)	
libeay32.dll ssleay32.dll	SSL utility libraries

You will need these named files for both 32-bit and 64-bit installations, the files have the same name in most cases, but the executables themselves are not the same. They will contain either 32-bit or 64-bit executable code, whichever is appropriate.

## SimbaClient for ODBC Required Files

You will need the following files to ensure that SimbaServer will work properly when installed at your customer's site:

Windows File	Description
SimbaClient.dll	The standard SimbaClient ODBC 3.52 driver.
SimbaClientConfig.cfg	
SimbaClientConnectionDialog.dll	
simbaicudt38_32.dll (simbaicudt38_64.dll if using 64-bit)	ICU utility libraries
simbaicu38_32.dll (simbaicu38_64.dll if using 64-bit)	
simbaicuuc38_32.dll (simbaicu38_64.dll if using 64-bit)	
libeay32.dll ssleay32.dll	SSL utility libraries

## Visual C++ 2005/2008 ODBC Redistributable Files

The Visual C++ 2005/2008 ODBC Redistributable files are a set of system files, distributed by Microsoft, that are required before ODBC drivers and connection will work. Many computers have these files installed already and there is no further requirement. However, if you experience fundamental problems connecting to ODBC data sources a simple first step is to install these redistributable files.

Microsoft licenses these files to be distributed and redistributed free of charge. You may install them on as many machines as you want with no restriction.

## SimbaClient for JDBC Required Files

SimbaClient for JDBC comes as a single .jar file. You can find it in the SimbaEngineSDK\8.1\DataAccessComponents\Lib folder in the installed SimbaEngine SDK. The name of the file is SimbaJDBCClient.jar. You do not need any other files to connect a JDBC Java application to SimbaServer.

## 4.2 Installation Procedure

This section contains the specific instructions for installing SimbaServer and SimbaClients for ODBC and JDBC on target machines.

### Windows Platforms

When you have gathered all the files you must install on your target, you can start to build your installer. The following steps describe what the installer must do to install that target.

#### Installing SimbaServer:

1. Determine where you will install SimbaServer on the target machine. You may install it in the system folder but it is probably better to have it installed in its own folder to prevent file collisions.
2. Copy all of the files in the SimbaServer Required Files list above to the target folder. Most of these files will be used from this folder. The vcredist\_x86.exe and vcredist\_x64.exe files are installers for the Visual C++ ODBC Redistributable files.
3. In the Setup folder, there are three server .reg files, SetupSimbaServer-32on32.reg, SetupSimbaServer-32on64.reg and SetupSimbaServer-64on64.reg. These contain examples of SimbaServer registry entries for the different installation scenarios. We have supplied these to help you understand what you need to do to properly configure SimbaServer on your customer's machines. There is also a ReadMe file in this folder that explains the details of the server configuration under each scenario. Follow the instructions in this ReadMe file.

4. When you have completed these steps on the target server, you can start SimbaServer from the command line or the Start -> Run command.

### Installing SimbaServer as a Windows Service:

You can run SimbaServer as a Windows service, which can be much more convenient for your customer because you can configure it to start automatically when Windows starts. Note that all commands need to be run from a command line that has administrative privileges or is 'Run as Administrator' (for Windows Vista or later). Here is how to do it:

1. Complete the installation of SimbaServer.
2. From a command line, execute "<Your DSI>SimbaServer.exe -Install". This will configure SimbaServer as a Windows Service on your local machine. If you want your installer to perform the configuration simply have it run the same command line.
3. The service as configured is called "SimbaService", and it is set up to start automatically when Windows starts. You can reconfigure it or manually start it immediately from the Services Administrative Control Panel. Alternatively you can also start and stop it from a command line by executing "net start SimbaService" to start it and "net stop SimbaService" to stop it.
4. To uninstall SimbaService, execute "<Your DSI>SimbaServer.exe -Uninstall" from a command prompt or have your installer execute the same command line.

### Installing SimbaClient:

You can install SimbaClient for ODBC either on the same machine as SimbaServer or on a different machine. If you install both on the same machine, the network software will simply loop back to the SimbaServer process and connect normally. This is a simpler setup for development and test than using two machines.

1. Determine where you will install SimbaClient for ODBC on the target machine. This might be in the system folder but it is probably better to have it installed in its own folder to prevent file collisions.
2. Copy all of the files in the SimbaClient for ODBC Required Files list, above, to the target folder. Most of these files will be used from this folder. The vcredist\_x86.exe and vcredist\_x64.exe files are installers for the Visual C++ ODBC Redistributable files.
3. In the Setup folder, there are three client .reg files, SetupSimbaClient-32on32.reg, SetupSimbaClient-32on64.reg and SetupSimbaClient-64on64.reg, These contain examples of the SimbaClient registry entries needed to install SimbaClient for ODBC. We have supplied these to help you understand what you need to do to properly configure SimbaClient on your customer's machines. There is also a ReadMe file in this folder that explains the details of how to configure the client. Follow the instructions in this ReadMe file.

## Testing the ODBC Client/Server Connection:

If you have successfully installed and started SimbaServer, you can test the connection from the client to the server:

1. Run a simple reporting application or ODBCtest from Microsoft.
2. Follow the steps to create a new ODBC connection in your test reporting application. Choose the default DSN created by your client installer. In the sample SimbaClient setup registry files, the name of the DSN created is SimbaODBCClientDSN. Click to request an ODBC connection using this DSN.
3. If your attempt to connect to SimbaServer is successful, try creating an SQL query and retrieving some data. This will make sure that the metadata and data functionality of the system is working as well.

## Installing and Testing the Java Client:

1. The Java Client requires that Java 1.5 or higher is installed. You can find a JDK at the Java website: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Before you proceed, please ensure that this is the version of Java used by your Java JDBC application.
2. Copy the SimbaJDBCClient.jar file to the target machine and make sure that it is on the java classpath.
3. The class name of the SimbaClient for JDBC to load is `com.simba.client.core.SimbaJDBCdriver` and a simple sample URL is "jdbc:simba://127.0.0.1".
4. We have supplied a simple program, JDBCsample.java, to test connectivity and to serve as an example JDBC application. The JDBCsample program is located in the `<InstallDir>\SimbaEngineSDK\8.1\Examples\JDBCsample` directory.
5. To test that SimbaClient for JDBC works properly, edit the JDBCsample.java program. Change the SQLQuery in JDBCsample.java to a query that is valid for your data store and DSI implementation. Ensure that the ServerPort, and ServerIP are both pointing at your SimbaServer machine. Running JDBCsample.java should print out the results of the executed SQL query. If you have not modified the query, you should see printed the contents of the "emp" table in the sample Quickstart database on the server.

## Conclusion

After successful installation, you can establish a connection between the universal ODBC and JDBC thin SimbaClient drivers and the SimbaServer.

## 5 Configuring SimbaServer

You can configure SimbaServer to tune the way it logs actions, transmits messages, manages resources and provides security. Configuring logging is often a matter of necessity when there is a problem you must diagnose. You usually know your security requirements in advance and you can configure SimbaServer once to meet them. Finding a good configuration for network communication and resource management is often a matter of modification and observation over time to find the best solution. This section explains the ways your users can configure SimbaServer to meet their needs. The following section explains how to configure SimbaClient.

Most of the SimbaServer configurations do not affect how you must configure SimbaClient because SimbaClient adapts to the SimbaServer configuration. However, there are a few exceptions where you have to make sure that you configure SimbaClient and SimbaServer to work together. These are highlighted and explained in this section and the following section. In all cases, it is best to make careful, stepwise changes to the configuration and test that SimbaClient can connect to SimbaServer after each one.

SimbaServer works and responds the same on Windows, UNIX and Linux. The configuration options are the same and they mean the same things. The only differences are in where the configuration options are stored.

### 5.1 SimbaServer on Windows

On Windows, the configuration information is stored in the registry. SimbaServer will look in the HKEY\_CURRENT\_USER key and then the HKEY\_LOCAL\_MACHINE key, in that order, to find its configuration settings. If you have configuration settings for SimbaServer under both keys, SimbaServer will stop looking after it finds any settings under the HKEY\_CURRENT\_USER key and will not look for any settings under the HKEY\_LOCAL\_MACHINE key.

SimbaServer looks for the sub-key called "SOFTWARE\Simba\Server" (or "SOFTWARE\Wow6432Node\Simba\Server for the 32-bit SimbaServer on a 64-bit machine) for its configuration settings. Each configuration section described below is a further sub-key in which the configuration values are stored. Thus, if the SimbaServer configuration is in the HKEY\_LOCAL\_MACHINE key, the value for setting the logging level will be:

```
\\HKEY_LOCAL_MACHINE\SOFTWARE\Simba\Server\Admin\LogLevel=LOG_OFF
```

We recommend that you create your configuration settings in the HKEY\_LOCAL\_MACHINE key for two reasons. First, if you run SimbaServer as a Windows service it will run by default under the System user ID. It is difficult to configure HKEY\_CURRENT\_USER registry values under the System user ID and while this difficulty can be solved in different ways, it is easier to avoid it completely. It is much easier to configure the values under the HKEY\_LOCAL\_MACHINE key, which is visible to all users. Second, if your primary configuration is under the HKEY\_LOCAL\_MACHINE key, you can do testing under a user ID and create an overriding

configuration in the HKEY\_CURRENT\_USER key of that user. This avoids having to change the HKEY\_LOCAL\_MACHINE key configuration until you know exactly what you want to do, and all other users IDs, including the System user ID, still see only the HKEY\_LOCAL\_MACHINE key configuration.

## 5.2 SimbaServer on UNIX and Linux

On UNIX and Linux machines, the configuration information is stored in a text file called `simbaserver.ini`. SimbaServer looks in several locations for this file. First, it looks in `<current working directory>/simbaserver.ini` (Note no leading period). Then it looks in `$HOME/.simbaserver.ini` (Note the leading period in the name in this case). Finally, it looks in `/etc/simbaserver.ini` (Note no leading period). If you have `simbaserver.ini` files in multiple locations, SimbaServer will stop looking after it finds any settings.

The `simbaserver.ini` file is divided into sections denoted by the section name in square brackets. Each section contains the keyword and value for each option set in that section. The keyword and value are of the form `keyword=value`. Each section ends with the title of the next section or with the end of the file. For example, the value for the logging level is set with a section in `simbaserver.ini` that looks like this:

```
[Admin]
LogLevel=LOG_OFF
<eof>
```

There is no reason for `simbaserver.ini` to be in one location or another. You can locate this file to suit your own needs.

## 5.3 SimbaServer Configuration Options

The SimbaServer configuration options are divided into sections. The sections are Admin, Buffers, Network and Threads. The sections are independent of each other.

Note that in the Windows Registry, integers can be stored as either a string or a dword value.

### Admin

Keyword	Description
LogLevel	How much information to log
UseSignaling	If OS Signals are handled in SimbaServer to shutdown cleanly

The Admin configuration section is for options that help the administration of the server.

<b>LogLevel</b>	How much information to log
<b>Required</b>	No
<b>Data type</b>	Enumeration
<b>Range</b>	LOG_OFF, LOG_FATAL, LOG_ERROR, LOG_WARNING, LOG_INFO, LOG_DEBUG, LOG_TRACE.
<b>Default value</b>	LOG_OFF
<b>Example</b>	LogLevel=LOG_ERROR
<b>Comment</b>	<p>With this keyword, you can control the amount of log output by controlling the kinds of events logged by SimbaServer.</p> <p>Possible values (case sensitive):            LOG_OFF: no logging occurs            LOG_FATAL: only log fatal errors            LOG_ERROR: log all errors            LOG_WARNING: log all errors and warnings            LOG_INFO: log all errors, warning, and informational messages            LOG_DEBUG: log method entry and exit points and parameter values for debugging            LOG_TRACE: log all method entry points.</p>

<b>UseSignaling</b>	If OS signals are handled in SimbaServer to shutdown cleanly
<b>Required</b>	No
<b>Data type</b>	Boolean
<b>Range</b>	0   1
<b>Default value</b>	1
<b>Example</b>	UseSignaling=1
<b>Comment</b>	This keyword specifies whether SimbaServer should handle signaled events or not (e.g. a 'Ctrl+C input). If enabled, SimbaServer will handle signaled events and shutdown cleanly.

## Buffers

Keyword	Description
BufferAllocationSize	Size of pre-allocated buffers for incoming and outgoing network messages
PreAllocatedBuffers	Number of pre-allocated buffers for incoming and outgoing network messages

The Buffers configuration section is for options that control the size and number of message buffers. These configuration options can affect both the performance and the resource management of the server.

<b>BufferAllocationSize</b>	Size of pre-allocated buffers for incoming and outgoing network messages
<b>Required</b>	No
<b>Data type</b>	Unsigned Integer (bytes)
<b>Range</b>	512 – sizeof (size_t)
<b>Default value</b>	8192
<b>Example</b>	BufferAllocationSize=4096
<b>Comment</b>	This keyword specifies the size of the message buffers that are pre-allocated when SimbaServer starts up. The minimum size you can specify is 512. Each of the <PreAllocatedBuffers> will be this big. SimbaServer uses these buffers to store incoming and outgoing network messages.

<b>PreAllocatedBuffers</b>	Number of pre-allocated buffer for incoming and outgoing network messages
<b>Required</b>	No
<b>Data type</b>	Unsigned Integer
<b>Range</b>	0 – 2 <sup>32</sup> -1
<b>Default value</b>	1000
<b>Example</b>	PreAllocatedBuffers=512
<b>Comment</b>	This keyword specifies the number of buffers to pre-allocate when SimbaServer starts up. Each buffer pre-allocated will be <BufferAllocationSize> bytes long. SimbaServer uses these buffers to store incoming and outgoing network messages.

## Network

Keyword	Description
ListenAddress	Bind the SimbaServer instance to a fixed IP address
ListenPort	The local port for SimbaServer to bind to
UseSsl	Control network security
SslCertfile	The SSL certificate file to use with SSL secure connections
SslKeyFile	The SSL private key file to use with SSL secure connections
ConnectionIdleTimeout	Connection idle timeout period
MaxConnections	The maximum number of connections allowed
MaxDataChunkSize	The maximum number of bytes of data in a single message

The Network configuration section is for options that control how SimbaClient and SimbaServer communicate. These configuration options affect both the performance and the resource management of the server.

<b>ListenAddress</b>	<b>Bind the SimbaServer instance to a fixed IP address</b>
Required	Yes
Data type	String
Range	Any valid IP address or hostname.
Default value	None.
Example	ListenAddress=192.168.0.2
Comment	This keyword restricts the server so it accepts TCP/IP connections only on the specified IP address. You will use it primarily in the case where a server has multiple network interface cards installed and where you want the server to accept connections only on a particular card. This is also known as a multi-homed server.

<b>ListenPort</b>	<b>The local port for SimbaServer to bind to</b>
Required	No
Data type	Signed 16-bit integer
Range	0-65535 (TCP/IP port number range.)
Default value	12345
Example	ListenPort=1583
Comment	This keyword specifies the port to which the server will bind and listen for TCP/IP connection requests. The range of the value is 0-65535.

<b>UseSsl</b>	<b>Control network security</b>
Required	No
Data type	Boolean
Range	0 or 1 (off or on)
Default value	0
Example	UseSsl=1
Comment	This keyword forces the connection between the SimbaClients and SimbaServer to use Secure Sockets Layer (SSL) security. If you turn on SSL security using this keyword, you will also have to specify a certificate file using the SslCertfile keyword, and specify a key file using the SslKeyFile keyword.

NOTE: If UseSsl is on, then UseSsl must also be turned on for SimbaClient. Conversely, if UseSsl is turned off on SimbaServer, it must also be turned off on SimbaClient.

<b>SslCertfile</b>	The SSL certificate file to use with SSL secure connections
<b>Required</b>	Yes, when UseSsl is on.
<b>Data type</b>	String
<b>Range</b>	Valid absolute or relative directory path to the SSL certificate file.
<b>Default value</b>	SimbaServerCertificate.pem
<b>Example</b>	SslCertFile=C:\SimbaServerCertificate.pem
<b>Comment</b>	This keyword specifies the full or relative path to the SSL certificate file to use with SSL secure connections. Note that the server will use this information only when you turn on SSL security with the UseSsl keyword.  OR:  This keyword points to the location of the certificate file for SimbaServer to use when encrypting SSL communication with SimbaClient.

<b>SslKeyFile</b>	The SSL private key file to use with SSL secure connections
<b>Required</b>	Yes, when UseSsl is True
<b>Data type</b>	String
<b>Range</b>	Valid absolute or relative directory path to the SSL server key file.
<b>Default value</b>	SimbaServerKey.pem
<b>Example</b>	SslKeyFile=C:\Simba Technologies\SimbaServerKey.pem
<b>Comment</b>	This keyword specifies the full or relative path to the SSL private key file to use with SSL secure connections. Note that the server will use this information only when you turn on SSL security with the UseSsl keyword.

<b>ConnectionIdleTimeout</b>	Connection idle timeout period
<b>Required</b>	No
<b>Data type</b>	Unsigned 32-bit integer (seconds)
<b>Range</b>	1 – 2 <sup>32</sup> -1 seconds until timeout
<b>Default value</b>	86400 (24 hours)
<b>Example</b>	ConnectionIdleTimeout=3600
<b>Comment</b>	This is the time that a connection can remain idle before SimbaServer will consider it timed out and will disconnect it. You can use this to prevent hung connections from clogging SimbaServer.

<b>MaxConnections</b>	The maximum number of total connections
<b>Required</b>	No
<b>Data type</b>	Unsigned 32-bit integer.
<b>Range</b>	0 – 2 <sup>32</sup> -1
<b>Default value</b>	512
<b>Example</b>	MaxConnections=256
<b>Comment</b>	This is the total number of connections that are permitted. This number includes both active and idle connections. Once this maximum number of connections is reached, subsequent connection requests will wait until one of the existing connections is disconnected. Refer to MaxWorkerThreads for the maximum number of active concurrent connections. If you expect a large number of users to access your server, this value should be set to a higher number.

<b>MaxDataChunkSize</b>	The maximum number of bytes of data in a single message
<b>Required</b>	No
<b>Data type</b>	Unsigned 16-bit integer
<b>Range</b>	1024 – 65535 bytes in a message.
<b>Default value</b>	65535
<b>Example</b>	MaxDataChunkSize=32768
<b>Comment</b>	This is the maximum number of bytes of data that SimbaServer will send to SimbaClient in a single message. SimbaServer may send fewer bytes in a single message if required. The actual number of bytes that SimbaServer may send to SimbaClient in a single message will range between 0 and MaxDataChunkSize, but you can't specify MaxDataChunkSize to be less than 1024.

## Threads

Keyword	Description
TaskQueueTimeout	Time to wait when attempting to queue a new request on the server
MinWorkerThreads	Number of Worker Threads created at startup
MaxWorkerThreads	Maximum number of active concurrent connections. Since each connection, when active, requires a worker thread to process its requests, the maximum number of worker threads is equal to the maximum number of active concurrent connections.

The Threads configuration section is specifically for options that control how SimbaServer manages Worker Threads. These configuration options can affect performance but are mostly to constrain resource (thread) use.

<b>TaskQueueTimeout</b>	Time to wait when attempting to queue a new request on the server.
<b>Required</b>	No
<b>Data type</b>	Unsigned 16-bit integer
<b>Range</b>	1 to 65536 milliseconds
<b>Default value</b>	1000
<b>Example</b>	TaskQueueTimeout=20000
<b>Comment</b>	This is the length of time SimbaServer will allow Worker Threads to wait before timing-out the request.

<b>MinWorkerThreads</b>	Number of Worker Threads created at startup
<b>Required</b>	No
<b>Data type</b>	Unsigned 16-bit integer
<b>Range</b>	0-65535 worker threads
<b>Default value</b>	10
<b>Example</b>	MinWorkerThreads=50
<b>Comment</b>	This is the number of Worker Threads that SimbaServer will create before starting up the server. SimbaServer will not allow the number of Worker Threads in the thread pool to dip below this number. Each active connection uses one worker thread to process its requests.

<b>MaxWorkerThreads</b>	Maximum number of active concurrent connections.
<b>Required</b>	No
<b>Data type</b>	Unsigned 16-bit integer
<b>Range</b>	0-65535 worker threads
<b>Default value</b>	64
<b>Example</b>	MaxWorkerThreads=32
<b>Comment</b>	<p>This is the maximum number of concurrent active connections that are permitted. NOTE that this is NOT the maximum number of connections allowed – i.e. there can be concurrent idle connections. Idle connections do not require the use of a worker thread.</p> <p>When this maximum number is reached, an active connection will wait until one of the active connections has had its requests serviced. Its worker thread will then be freed for use by this active connection.</p> <p>If you expect a large number of concurrently active users, this number should be set higher. This number <b>_MUST NOT_</b> exceed MaxConnections.</p> <p>Refer to MaxConnections for the maximum total number of connections permitted.</p>

## 6 Configuring SimbaClient

You configure SimbaClient through the keywords and values in each DSN (for ODBC) or connection string (for ODBC and JDBC). This means that there is no central location for configuration information on user machines like there is for SimbaServer. Nevertheless, you can configure SimbaClient to tune the way it logs actions and transmits messages. You must match the security configuration to SimbaServer. Of course, you must also specify how to find SimbaServer.

SimbaClient works and responds the same on Windows, UNIX and Linux. The configuration options are the same and they mean the same things. The only differences are in where the configuration options are stored.

### 6.1 SimbaClient for ODBC

You configure SimbaClient for ODBC with keywords and values in the DSN for each database. You can use the same keywords and values for System, User and File DSNs, and in .ini files on UNIX and Linux. The application can also dynamically add configuration keywords and values to the connection string. This may not be useful for commercial applications but it can be very useful for custom applications.

#### SimbaClient for ODBC on Windows

On Windows, you will most frequently configure SimbaClient for ODBC by adding keywords and values to each DSN. On Windows, there are three kinds of DSN: System DSNs, which are visible to all users, User DSNs, which are visible to a specific user, and File DSNs, which are also visible to a specific user but can be shared by copying the DSN file.

#### Configuring ODBC DSNs on Windows

On Windows, each ODBC driver typically has a configuration DLL that allows you to configure DSNs with a dialog box rather than directly by editing the Windows Registry. Using the configuration dialog is safer than editing the Windows Registry because the logic in the dialog prevents you from saving an inconsistent configuration that hampers driver operation. SimbaClient for ODBC has its own configuration dialog and we recommend that you use it.

To use the SimbaClient for ODBC configuration dialog, you must have SimbaClient for ODBC properly installed on your system. Click on Start -> Control Panel -> Administrative Tools -> Data Sources (ODBC) to open the ODBC Data Source Administrator. You can either click on Add... to create a new data source and select SimbaClient from the list of ODBC drivers, or select a DSN configured to use SimbaClient for ODBC and click on Configure... . This will open the SimbaClient for ODBC configuration dialog on the DSN, or open a blank DSN.

On the main tab of the configuration dialog, you will see several of the DSN configuration options in the GUI, such as Server IP and Server Port. All SimbaClient DSNs need these entries so they can connect to SimbaServer. If you click on the Option, Logging and Secondary Servers buttons, you will be able to see all of the other configuration options used in the DSN. If you want to use a configuration keyword in a DSN that does not have an equivalent entry in the configuration dialog, you can add it to the Custom Property list that is accessible via the Options button. The configuration dialog will write the keyword and value into the DSN when you click OK. You may want to use these keywords for fine-tuning and troubleshooting, but it is unlikely that end users would want to use them. We have chosen not to make them easily visible to end users.

## SimbaClient for ODBC on UNIX and Linux

On UNIX and Linux, you will most frequently configure SimbaClient for ODBC by adding keywords and values to each DSN. On UNIX and Linux there is only one kind of DSN and it is stored in a text file called `odbc.ini`. There are several locations where SimbaClient for ODBC will look for this file. First, it looks in `$ODBCSYSINI/odbc.ini` (Note no leading period). Then, it looks in `$HOME/.odbc.ini` (Note the leading period in the name in this case). Finally, it looks in `/etc/odbc.ini` (Note no leading period). If you have `odbc.ini` files in more than one location, SimbaClient will stop looking after it finds the file in the first directory (in the order specified above).

The `odbc.ini` file is divided into sections corresponding to the DSNs. Each section is denoted by the DSN name in square brackets. Each DSN contains the keyword and value for each option set for that DSN. The keyword and value are of the form `keyword=value`. Each DSN section ends with the title of the next DSN or with the end of the file. For example, the value for the logging level is set with a section in `simbserver.ini` that looks like this:

```
[SimbaODBCClientDSII]
SERVERS=192.168.0.1 12345,192.168.0.2 1583
LogLevel=LOG_OFF
<eof>
```

There is no configuration dialog for UNIX or Linux. You will have to edit the `odbc.ini` file with a text editor.

## SimbaClient for ODBC Configuration Options

The following table lists all of the keywords understood by SimbaClient. The subsections following the table explain in detail each keyword and its acceptable values.

Keyword	Description
Description	A brief, human-readable description of the DSN
Driver	The location of the driver file
ConnectionDialog	The location of the connection dialog file.
Servers	A listing of all servers to connect to.
ServerIp <b>**DEPRECATED**</b> (See Note)	The IP address of SimbaServer.
ServerPort <b>**DEPRECATED**</b> (See Note)	The port on which SimbaServer is listening.
ConnectionTimeout	The time to wait for a response on the connection
MsgFetchSize	The number of messages to request for each read data request.
PushedParamCacheSize	Parameter cache size for SQL_DATA_AT_EXEC.
EnableLogging	Turn troubleshooting logging on or off.
LogDirectory	Directory to store log files in.
LogNamespace	The namespace to log.
LogLevel	How much information to log.
LogMilliseconds	Include timestamp in log records.
LogThreadId	Include the thread id in log records.
UseSsl	Enable SSL network encryption.
SSLCertfile	Location of the SSL certificate file.

Note **\*\*DEPRECATED\*\***: The ServerIp and ServerPort entries are now deprecated. They will still work and act as the primary server, but will be removed from the DSN when the configuration dialog is used.

Note that in the Windows Registry, integers can be stored as either a string or a dword value.

Description	A brief, human-readable description of the DSN
Required	No
Data type	String (DSN name)
Range	N/A
Default value	Sample SimbaODBCClient DSN
Example	Description=Time system database – read-only.
Comment	This describes the DSN to users who are deciding which DSN to use. Applications will usually show the description alongside the DSN in a list from which the user chooses.

<b>Driver</b>	The location of the ODBC driver
<b>Required</b>	No, but nice to have.
<b>Data type</b>	String (path)
<b>Range</b>	N/A
<b>Default value</b>	None
<b>Example</b>	Driver= C:\Simba Technologies\...\SimbaClient.dll
<b>Comment</b>	The Driver keyword points to the location of the ODBC driver itself. This is not always required as the location can be inferred, but it is safest to have it.

<b>ConnectionString</b>	The location of the connection dialog driver
<b>Required</b>	Only if the connection dialog is required
<b>Data type</b>	String (path)
<b>Range</b>	N/A
<b>Default value</b>	<InstallDir>\ConnectionString.dll
<b>Example</b>	ConnectionString=C:\Simba Technologies\...\ConnectionString.dll
<b>Comment</b>	You can use the ConnectionDialog.dll for a staged login when you need to have the client and server have a short conversation before connecting.

<b>Servers</b>	A listing of all the servers to connect to
<b>Required</b>	Yes
<b>Data type</b>	String
<b>Range</b>	IP addresses: Any valid IP address. Ports: 0 - 65535
<b>Default value</b>	127.0.0.1 12345 (loopback with default port)
<b>Example</b>	SERVERS=127.0.0.1 12345,192.168.0.1 12345,
<b>Comment</b>	SimbaClient must be able to find SimbaServer on the network. This keyword enables it. SimbaClient will try to make a network connection to the servers in the order specified until a connection is made. The format is: <ip> <port>,<ip> <port>,...

<b>ServerIp</b>	The IP address of SimbaServer
<b>Required</b>	<b>**Note: DEPRECATED**</b>
<b>Data type</b>	String
<b>Range</b>	IP addresses: Any valid IP address. Ports: 0 - 65535
<b>Default value</b>	127.0.0.1 (loopback)
<b>Example</b>	ServerIp=192.168.0.1
<b>Comment</b>	SimbaClient must be able to find SimbaServer on the network. This keyword enables it. SimbaClient will try to make a network connection to this address.

<b>ServerPort</b>	The port on which SimbaServer is listening
<b>Required</b>	<b>**Note: DEPRECATED**</b>
<b>Data type</b>	Unsigned Integer (16-bit)
<b>Range</b>	0 - 65535
<b>Default value</b>	12345
<b>Example</b>	ServerPort=1583
<b>Comment</b>	SimbaClient must be able to make a network connection to the IP address and port on which SimbaServer is listening. This keyword supplies the port number. SimbaClient will try to make a network connection to this port number on the ServerIp address.

<b>ConnectionTimeout</b>	SimbaClient connection request timeout
<b>Required</b>	No
<b>Data type</b>	Unsigned integer
<b>Range</b>	0 – INT_MAX seconds
<b>Default value</b>	0 (Indefinitely)
<b>Example</b>	ConnectionTimeout=60000
<b>Comment</b>	SimbaClient will wait indefinitely for SimbaServer to respond to a network connection unless you specify a timeout period. ConnectionTimeout specifies how many seconds that SimbaClient will wait before aborting the connection attempt and returning to the application with an error.

<b>MsgFetchSize</b>	The number of messages to request from the server for each read data request
<b>Required</b>	No
<b>Data type</b>	Unsigned 16-bit integer
<b>Range</b>	0 – 65535
<b>Default value</b>	10
<b>Example</b>	MsgFetchSize=10
<b>Comment</b>	SimbaClient requests messages from SimbaServer in groups to reduce network overhead. This keyword acts as a hint for the number of messages to return, but will not necessarily be the actual number returned.

<b>PushedParamCacheSize</b>	Size of the parameter cache for SQL_DATA_AT_EXEC
<b>Required</b>	No
<b>Data type</b>	Unsigned integer
<b>Range</b>	0 – INT_MAX bytes. Any value below 128 will turn this function off.
<b>Default value</b>	8192
<b>Example</b>	PushedParamCacheSize=16384
<b>Comment</b>	This keyword specifies the number of bytes set aside to cache input parameter data that has been designated SQL_DATA_AT_EXEC. If you turn off this function, each parameter passed from the application will be immediately sent to the server. This is usually very inefficient and slows down processing.

<b>EnableLogging</b>	Globally control logging
<b>Required</b>	No
<b>Data type</b>	Boolean
<b>Range</b>	false   true no   yes y   n t   f 0   1 or any non-zero
<b>Default value</b>	0 (off)
<b>Example</b>	EnableLogging=true
<b>Comment</b>	This keyword allows you to turn logging on and off without changing the logging parameters specified elsewhere. It is handy to see how changes affect the client without making it inconvenient to restart logging.

<b>LogDirectory</b>	Location for log files.
<b>Required</b>	No
<b>Data type</b>	String
<b>Range</b>	Valid directory path or blank
<b>Default value</b>	Blank. This stores log files in the current working directory.
<b>Example</b>	LogDirectory=C:\Simba Technologies\Temp
<b>Comment</b>	This keyword specifies where SimbaClient will write all log files. If no path is specified then the log files will be written to the current working directory.

<b>LogNamespace</b>	The SimbaClient code namespace to log
<b>Required</b>	No
<b>Data type</b>	String
<b>Range</b>	All valid SimbaClient namespaces or blank
<b>Default value</b>	Simba. This logs all namespaces.
<b>Example</b>	LogNamespace=Simba
<b>Comment</b>	With this keyword, you can narrow the areas of code in SimbaClient that write information to log files. This is useful if you know a problem is in one area and you do not want to have all your log files cluttered with information from other namespaces that do not interest you.
<hr/>	
<b>LogLevel</b>	Set the logging verbosity
<b>Required</b>	No
<b>Data type</b>	Enumeration
<b>Range</b>	LOG_OFF, LOG_FATAL, LOG_ERROR, LOG_WARNING, LOG_INFO, LOG_DEBUG, LOG_TRACE
<b>Default value</b>	LOG_OFF
<b>Example</b>	LogLevel=LOG_ERROR
<b>Comment</b>	With this keyword, you can control the amount of log output by controlling the kinds of events that are logged.  Possible values (case sensitive): LOG_OFF: no logging occurs LOG_FATAL: only log fatal errors LOG_ERROR: log all errors LOG_WARNING: log all errors and warnings LOG_INFO: log all errors, warning, and informational messages LOG_DEBUG: log method entry and exit points and parameter values for debugging LOG_TRACE: log all method entry points.
<hr/>	
<b>LogMilliseconds</b>	Include time in milliseconds on each log line
<b>Required</b>	No
<b>Data type</b>	Boolean
<b>Range</b>	0   1
<b>Default value</b>	1
<b>Example</b>	LogMilliseconds=0
<b>Comment</b>	Turning this keyword on adds a time stamp to the millisecond to each log output line. This can be useful for seeing where your system is taking unexpected amounts of time.

<b>LogThreadld</b>	Include the thread ID in each log line
<b>Required</b>	No
<b>Data type</b>	Boolean
<b>Range</b>	0   1
<b>Default value</b>	1
<b>Example</b>	LogThread=0
<b>Comment</b>	Turning this keyword on allows you to separate log lines by thread ID and better understand what the system is doing.
<b>UseSsl</b>	Enable SSL network encryption
<b>Required</b>	Yes if UseSsl is enabled on the Server.
<b>Data type</b>	Boolean
<b>Range</b>	0   1 (off or on)
<b>Default value</b>	0
<b>Example</b>	UseSsl=1
<b>Comment</b>	When SimbaServer has SSL network encryption enabled, SimbaClients that want to connect to it must also have SSL enabled. SimbaClient and SimbaServer use OpenSSL encryption to encrypt all data moving across their network connections.
<b>SSLCertfile</b>	Location of the SSL certificate file
<b>Required</b>	Yes if UseSsl is enabled
<b>Data type</b>	String
<b>Range</b>	Valid absolute or relative directory path to the SSL certificate file.
<b>Default value</b>	SimbaCACertificate.pem
<b>Example</b>	SSLCertificateFile= C:\Simba Technologies\SimbaEngineSDK\8.1\SSLCertificates\SimbaCACertificate.pem
<b>Comment</b>	This keyword points to the location of the certificate file for SimbaClient to use when encrypting SSL communication with SimbaServer.

## 6.2 SimbaClient for JDBC

SimbaClient for JDBC responds to configuration keywords similar to those for the SimbaClient for ODBC. However, since JDBC does not use the idea of a DSN, the user application must add the keywords and values to the connection string. SimbaClient for JDBC parses the connection string for configuration keywords, and extracts and acts on the values.

The general form of the connection URL is:

*`jdbc:simba://<HOST>[:<ServerPort>][/<ServerDSN>][;<property>=<value>[;...]]`*

where [ ] indicates optional values. Note that additional values that need to be passed through the client, such as UID and PWD, can be specified in the property=value portion of the URL.

Keyword	Description
HOST	
ServerPort	The port on which SimbaServer is listening
SERVERS	A list of secondary servers.
ConnectionTimeout	The time to wait for a server response
ServerDSN	Data Source Name for a SimbaD2O Server to connect to
UseSsl	Enable SSL network encryption
TrustedStorePath	The location of the java keystore
TrustedStorePassword	The password used to access the trusted key store
MsgFetchCount	The number of messages to request from the server for each read data request
LogLevel	How much information to log
LogDirectory	Directory to store log files in

<b>HOST</b>	<b>The hostname or IP address of SimbaServer.</b>
<b>Required</b>	Yes
<b>Data type</b>	String
<b>Range</b>	Any valid hostname or IP address
<b>Default value</b>	(No default)
<b>Example</b>	192.168.0.1 or [2001:0db8:85a3:0000:0000:8a2e:0370:7334]
<b>Specification</b>	This value must be present in the connection URL. Example: <i><code>jdbc:simba://&lt;HOST&gt;</code></i> where <i><code>&lt;HOST&gt;</code></i> is replaced with the appropriate value. Note that the use of [ ] is optional for IPv4 addresses, but required for IPv6 addresses.
<b>Comment</b>	SimbaClient must be able to find SimbaServer on the network and will try to make a network connection to this hostname or address. If the connection fails and SERVERS (see the corresponding section) is also specified, then SimbaClient will continue on to attempt a connection to each of the secondary servers.

<b>ServerPort</b>	The port on which SimbaServer is listening
<b>Required</b>	No
<b>Data type</b>	Integer
<b>Range</b>	0 – 65535
<b>Default value</b>	12345
<b>Example</b>	ServerPort=1583
<b>Specification</b>	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;:&lt;ServerPort&gt;</i> where the values in < > are replaced with the appropriate values.
<b>Comment</b>	SimbaClient must be able to make a network connection to the host and port on which SimbaServer is listening. SimbaClient will try to make a network connection to this port number.

<b>SERVERS</b>	A list of secondary servers.
<b>Required</b>	No
<b>Data type</b>	String
<b>Range</b>	Host: Any valid hostname or IP address Port: 0 – 65535
<b>Default value</b>	(No default)
<b>Example</b>	SERVERS=192.168.0.1:12345,[2001:0db8:85a3:0000:0000:8a2e:0370:7334]:1583
<b>Specification</b>	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;[...];SERVERS=&lt;ip&gt;:&lt;port&gt;[,&lt;ip&gt;:&lt;port&gt;...]</i> where the values in < > are replaced with the appropriate values and [ ] indicates optional portions of the connection URL.
<b>Comment</b>	SimbaClient must be able to find SimbaServer on the network. If the connection fails using the primary server (as specified by the HOST and ServerPort), then SimbaClient will continue on to attempt a connection to each of the secondary servers specified in order until a connection is made.

<b>ConnectionTimeout</b>	The time to wait for a server response
<b>Required</b>	No
<b>Data type</b>	Integer
<b>Range</b>	0 – INT_MAX milliseconds
<b>Default value</b>	0 (Indefinitely)
<b>Example</b>	ConnectionTimeout=60000
<b>Specification</b>	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;[...];ConnectionTimeout=&lt;TimeoutValue&gt;</i> where the values in < > are replaced with the appropriate values and [ ] indicates optional portions of the connection URL.
<b>Comment</b>	SimbaClient will wait indefinitely for SimbaServer to respond to a network connection unless you specify a timeout period. ConnectionTimeout specifies how many milliseconds that SimbaServer will wait before aborting the connection attempt and returning to the application with an error.

<b>ServerDSN</b>	<b>Data Source Name for a SimbaD2O Server to connect to</b>
<b>Required</b>	Only if the server is a SimbaD2O Server
<b>Data</b>	String
<b>Range</b>	Any valid DSN on the host machine that the SimbaD2O server may connect to.
<b>Default value</b>	(No default)
<b>Example</b>	ServerDSN=QuickstartDSII
<b>Specification</b>	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;[...]/&lt;ServerDSN&gt;</i> where the values in < > are replaced with the appropriate values and [ ] indicates optional portions of the connection URL.
<b>Comment</b>	
<b>UseSsl</b>	<b>Enable SSL network encryption</b>
<b>Required</b>	Yes if UseSsl is enabled on the Server.
<b>Data type</b>	String
<b>Range</b>	Off: 0, f, false, n, or no . On: 1, t, true, y, or yes .
<b>Default value</b>	0
<b>Example</b>	UseSsl=1
<b>Specification</b>	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;[...];UseSsl=&lt;SslValue&gt;</i> where the values in < > are replaced with the appropriate values and [ ] indicates optional portions of the connection URL.
<b>Comment</b>	When SimbaServer has SSL network encryption enabled, SimbaClients that want to connect to it must also have SSL enabled. SimbaClient and SimbaServer use OpenSSL encryption to encrypt all data moving across their network connections.
<b>TrustedStorePath</b>	<b>The location of the java keystore</b>
<b>Required</b>	Yes if UseSsl is enabled
<b>Data type</b>	String
<b>Range</b>	Valid absolute or relative directory path to the SSL trusted store.
<b>Default value</b>	(No default)
<b>Example</b>	TrustedStorePath="C:\JDBCKeyStore"
<b>Specification</b>	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;[...];UseSsl=1;TrustedStorePath=&lt;Path&gt;</i> where the values in < > are replaced with the appropriate values and [ ] indicates optional portions of the connection URL.
<b>Comment</b>	A TrustedStore must be created on the Client if UseSsl is enabled on the Server.

<b>TrustedStorePassword</b>	<b>The password used to access the trusted key store</b>
Required	Yes if UseSsl is enabled
Data type	String
Range	Valid trusted store password.
Default value	(No default)
Example	TrustedStorePassword=SimbaServer12345
Specification	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;[...];UseSsl=1;TrustedStorePath=&lt;Path&gt;;TrustedStorePassword=&lt;Password&gt;</i> where the values in < > are replaced with the appropriate values and [ ] indicates optional portions of the connection URL.
Comment	A TrustedStore must be created on the Client if UseSsl is enabled on the Server.

<b>MsgFetchCount</b>	<b>The number of messages to request from the server for each read data request</b>
Required	No
Data type	Unsigned 16-bit integer
Range	0 - 65535
Default value	50
Example	MsgFetchCount=50
Specification	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;[...];MsgFetchCount=&lt;FetchValue&gt;</i> where the values in < > are replaced with the appropriate values and [ ] indicates optional portions of the connection URL.
Comment	SimbaClient requests messages from SimbaServer in groups to reduce network overhead. This keyword acts as a hint for the number of messages to return, but will not necessarily be the actual number returned.

<b>LogLevel</b>	How much information to log
<b>Required</b>	No
<b>Data type</b>	Enumeration
<b>Range</b>	OFF or 0, FATAL or 1, ERROR or 2, WARNING or 3, INFO or 4, DEBUG or 5, TRACE or 6
<b>Default value</b>	OFF (0)
<b>Example</b>	LogLevel=DEBUG
<b>Specification</b>	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;[...];LogLevel=&lt;LevelValue&gt;</i> where the values in < > are replaced with the appropriate values and [ ] indicates optional portions of the connection URL.
<b>Comment</b>	With this keyword, you can control the amount of log output by controlling the kinds of events that are logged.  Possible values: OFF (0): no logging occurs FATAL (1): only log fatal errors ERROR (2): log all errors WARNING (3): log all errors and warnings INFO (4): log all errors, warnings, and informational messages DEBUG (5): log method entry and exit points and parameter values for debugging TRACE (6): log all method entry points.
<b>LogDirectory</b>	Directory to store log files in
<b>Required</b>	No
<b>Data type</b>	String
<b>Range</b>	Valid directory path or blank
<b>Default value</b>	Blank. This stores log files in the current working directory.
<b>Example</b>	LogDirectory="C:\Simba Technologies\Temp"
<b>Specification</b>	This value may be present in the connection URL. Example: <i>jdbc:simba://&lt;HOST&gt;[...];LogLevel=&lt;NotOff&gt;;LogDirectory=&lt;DirectoryValue&gt;</i> where the values in < > are replaced with the appropriate values and [ ] indicates optional portions of the connection URL.
<b>Comment</b>	This keyword specifies where SimbaClient will write all log files. If no path is specified then the log files will be written to the current working directory.

## 7 Configuring Secure Sockets Layer (SSL)

SimbaClient/Server supports Secure Sockets Layer (SSL) security between SimbaClient and SimbaServer connections. Refer to the Configuring SimbaServer and Configuring SimbaClient sections for instructions on how to enable SSL. If SSL is enabled, SimbaClients and SimbaServer will use OpenSSL encryption to encrypt all data moving across their network connections.

This section contains information about working with SSL.

Example SSL certificates have been supplied in the <InstallDir>\SimbaEngineSDK\8.1\SSLCertificates directory. These are Simba self-signed certificates that were created using OpenSSL.

To enable SSL for SimbaServer and Simba Client, a set of SSL certificates must be generated. The idea is that the Certificate Authority (CA) certificate that was used to sign the Server Certificate becomes the 'SslCertfile' that the Client will use to authenticate the Server.

### 7.1 Generating a Certificate Authority (CA) Certificate –for self-signing

This section covers the process of establishing yourself as a root certificate authority for self-signing your certificates. Self-signed certificates are particularly useful during the development and testing phases. This allows you to avoid spending unnecessary money on getting your certificates signed until you are ready for production. Once the testing phase is over, and you are ready to generate a commercially-signed certificate, proceed to the Generating an SSL Certificate with Verisign section.

First, install OpenSSL. You can find OpenSSL at <http://openssl.org>.

Add the path to the openssl.exe executable to your PATH variable. Refer to <http://openssl.org> for other configuration options and/or values.

#### Create the Root CA Certificate

1. Create the directory C:\newcerts.  
`md C:\newcerts`
2. Change to the *newcerts* directory.  
`cd C:\newcerts`
3. Generate a CA private key.

```
openssl genrsa -out CA-key.pem 2048
```

4. Generate the root CA certificate.

```
openssl req -new -key CA-key.pem -x509 -days 1000 -out CA-  
cert.pem
```

You will be asked a series of questions which will be incorporated into the certificate, such as Country, City, Company Name, etc. Remember what information you entered as you may get prompted for this information again at a later stage. When asked for an email address, provide the email address of the CA contact.

You will need *CA-key.pem* and *CA-cert.pem* in the following steps.

## Creating and Signing a Server Certificate

1. Generate a new key:

```
openssl genrsa -out server-key.pem 2048
```

2. Generate a certificate signing request. The *openssl.cnf* file can be found in your OpenSSL installation directory. Copy it to the *newcerts* directory. You may need to modify some of the configuration settings in this file. Enter the following command (all in one line).

```
openssl req -new -config openssl.cnf -key server-key.pem -out  
signingReq.csr
```

3. Self-sign the certificate using your *CA-cert.pem* certificate. Enter the following command (all in one line).

```
openssl x509 -req -days 365 -in signingReq.csr -CA CA-cert.pem -  
CAkey CA-key.pem -CAcreateserial -out server-cert.pem
```

## 7.2 Generating an SSL Certificate with Verisign

First, install OpenSSL. You can find OpenSSL at <http://openssl.org>.

Add the path to the `openssl.exe` executable to your PATH variable. Refer to <http://openssl.org> for other configuration options and/or values.

### Create the Server Private Key

1. Create the directory `C:\newcerts`

```
md C:\newcerts
```

2. Change to the *newcerts* directory.

```
cd C:\newcerts
```

3. Generate a new key:

```
openssl genrsa -out server-key.pem
```

4. Generate a certificate signing request.

```
openssl req -new -key server-key.pem -out signingReq.csr
```

You will be asked a series of questions which will be incorporated into the certificate request, such as Country, City, Company Name, etc. When asked for an email address, provide a valid email address because Verisign will send you the signed certificate via this email address.

**Note:** The information you enter will be verified when sending the request to a trusted authority.

5. Send the request (*signingReq.csr*) to the Certificate Authority (Verisign). You may need to verify that the information collected when generating *signingReq.csr* is correct.
6. If successful, the Certificate Authority (Verisign) will send you a certificate, via the email address you provided. In the email, there will be an encrypted CA certificate data and a link to an encrypted CA intermediate certificate data. Copy both certificate data to a text file, with the non-intermediate certificate data followed by the intermediate certificate data. This text file will be referred to as *CA-cert.pem* in the following steps.

## Creating and Signing a Server Certificate

This section requires *server-key.pem*, *signingReq.csr*, and *CA-cert.pem*, which you generated in the previous section.

1. Copy the *CA-cert.pem* file to your *C:\newcerts* directory. Ensure the *server-key.pem*, and *signingReq.csr* files are in this directory as well.
2. Change to the *newcerts* directory.

```
cd C:\newcerts
```

3. Create the server certificate. Enter the following command (all in one line).

```
openssl CA -in signingReq.csr -out server-cert.pem -keyfile  
server-key.pem -days 365 -cert CA-cert.pem
```

## 7.3 Creating a Trusted Key Store for JDBCClient

In order to enable SSL for the JDBCClient, a Java keystore must be created. The Java keystore specifies a certificate storage location.

To create a trusted store, first ensure that the java bin directory is in your PATH variable. Open a cmd window and type:

```
keytool -import -alias "JDBCKeyStore" -file <path_to>\CA-cert.pem -
keystore C:\JDBCKeyStore
```

You will be prompted for a password. You'll also be asked to verify that the given certificate should be trusted. Type 'yes' when prompted.

Note: The `-keystore` path you specified will be the value you use for the 'TrustedStorePath' keyword. The password you specified for your keystore will be the value you use for the 'TrustedStorePassword' keyword.

An example connection URL, which enables SSL is:

```
jdbc:simba://localhost;UseSsl=1;TrustedStorePath=C:\\JDBCKeyStore;Trus
tedStorePassword=test
```

## 7.4 Distributing SSL Certificates

You should now have a full set of SSL certificates ready for distribution. Refer to the instructions in [Generating a Certificate Authority \(CA\) Certificate –for self-signing](#) or [Generating an SSL Certificate with Verisign](#) for generating SSL certificates if you do not.

You should have three certificates files: a server key file, a server certificate file, and a Certificate Authority (CA) file.

- Server Key File (e.g. *server-key.pem*) – this will be the file for the 'SSLKeyFile' configuration keyword for SimbaServer.
- Server Certificate File (e.g. *server-cert.pem*) – this will be the file for the 'SSLCertfile' configuration keyword for SimbaServer.
- Certificate Authority File (e.g. *CA-cert.pem*) – this will be the file for the 'SSLCertfile' configuration keyword for SimbaClient. For JDBCClient, this will be the certificate file that needs to be added to the Java keystore.

## 8 Enabling Logging

### 8.1 SimbaServer

The logging in SimbaServer is controlled by several settings:

1. **LogLevel:** This setting controls the verbosity of the logger. To allow for the creation of log files, this should be set to a level greater than LOG\_OFF.
2. **DSILogging:** This setting is the global switch that turns on or off logging completely. A value of '1' will enable logging while a value of '0' will disable it.
3. **DSILoggingPath:** This is an optional setting that controls where the created log file is located. If this is not set, the default location is the current working directory of the SimbaServer (note that this usually differs when the SimbaServer is running as an executable versus a service).

How these configuration options are set differs on Windows versus UNIX and Linux.

#### Windows

The LogLevel can be set as described in section 5.1 SimbaServer on Windows.

DSILogging and DSILoggingPath are registry entries that are located under the HKEY\_LOCAL\_MACHINE \Software\Simba\Driver key (or for a 32-bit driver on a 64-bit system, under the HKEY\_LOCAL\_MACHINE \Software\Wow6432Node\Simba\Driver key).

#### UNIX and Linux

The LogLevel can be set as described in section 5.2 SimbaServer on UNIX and Linux.

DSILogging and DSILoggingPath are entries that can be set in the simba.ini file. The search order for this file is as follows:

1. If the SIMBAINI environment variable is defined, then the value of SIMBAINI is used to locate the file. SIMBAINI must contain the full path including the filename.
2. Otherwise, if the SIMBAINI environment variable is not defined, the current working directory of the application will be searched for simba.ini. (Note the lack of a preceding 'dot'.)
3. The next directory that will be searched is \$HOME for .simba.ini. (Note the preceding 'dot'.)
4. Finally, the system wide default /etc/simba.ini will be used. (Note the lack of a preceding 'dot'.)

In the [Driver] section, add the DSILogging and DSILoggingPath entries. For example:

```
[Driver]
DSILogging=1
```

```
DSILoggingPath=/abc/logs
```

## 8.2 SimbaClient for ODBC

The logging in SimbaClient for ODBC is controlled by several settings:

1. **LogLevel:** This setting controls the verbosity of the logger. To allow for the creation of log files, this should be set to a level greater than LOG\_OFF.
2. **EnableLogging:** This setting is the global switch that turns on or off logging completely. A value of '1' will enable logging while a value of '0' will disable it.
3. **LogDirectory:** This setting controls where the created log file is located. If this is not set, the default location is the current working directory of the application using the SimbaClient for ODBC.
4. **LogNamespace, LogMilliseconds, and LogThreadId:** These settings fine-tune what is being recorded in the log file.

These configuration settings can be set as described in 6.1 SimbaClient for ODBC.

## 8.3 SimbaClient for JDBC

The logging in SimbaClient for JDBC is controlled by two settings:

1. **LogLevel:** This setting controls the verbosity of the logger. To allow for the creation of log files, this should be set to a level greater than LOG\_OFF.
2. **LogDirectory:** This is an optional setting that controls where the created log file is located. If this is not set, the default location is the current working directory of the application using the SimbaClient for JDBC.

Both configuration settings can be set as described in 6.2 SimbaClient for JDBC.

## 9 Advanced Information

This section contains advanced information about the architecture and operation of SimbaClient/Server. You probably do not need to know this information to get your first SimbaServer running properly. However, as you improve your DSI implementation and get ready to ship your product to customers, you will probably benefit from reading this section to gain a deeper understanding of how the client and server components work and how they can be made more reliable.

### 9.1 SimbaServer Architecture

SimbaServer is a single process that manages many separate threads to deliver high performance, remote access to your data.

#### Start Up

When SimbaServer starts, it reads the registry to find the port to which it must bind. It then checks to see if another instance of SimbaServer is already running and bound to that port. If there is already another process bound to the port, the SimbaServer process will shut down and refuse to run. An appropriate error message is displayed. Once the SimbaServer process is started and bound to its port, it begins listening for connection requests from SimbaClients.

#### SimbaServer Overview

In the SimbaServer process, a single specialized thread, called the Listening Thread, accepts the incoming TCP/IP connection requests from SimbaClients. The Listening Thread passes the TCP/IP connection requests to a second thread, the Select Thread. The Select Thread reads the ODBC request messages from active connections and puts them into a queue. SimbaServer assigns each TCP/IP connection to a single Worker Thread. It maintains inactive Worker Threads in a pool. The Worker Thread continues to service the messages arriving on that TCP/IP connection until SimbaServer drops the connection. Then, the Worker Thread goes back into the pool. Each active Worker Thread reads its ODBC requests from the queue and translates it into a series of method calls into the DSI implementation. The results of the method calls are processed and returned to the SimbaClient in a TCP/IP message.

#### Listening Thread

SimbaServer uses a single Listening Thread to accept TCP/IP connection requests from SimbaClients. As a security option, you can configure the Listening Thread to accept incoming connection requests only from a specific IP address. The Listening Thread will accept a maximum of 1024 TCP/IP connections. When it reaches this limit, it responds to subsequent connection requests with a TCP/IP error until some connections are dropped. You cannot configure this connection limit at run time.

## Select Thread

The Select Thread monitors all of the TCP/IP connections for incoming requests. It reads the requests from all of the SimbaClients, builds tasks that encapsulate the request messages and then queues the tasks. The Worker Thread dedicated to the TCP/IP connection retrieves the queued tasks and executes them.

## Worker Pool

The Worker Pool contains a fluctuating number of Worker Threads, depending on server activity. At start-up, SimbaServer creates a number of Worker Threads to seed the pool. The initial number of Worker Threads in the Worker Pool is configurable. If the number of TCP/IP connections empties the Worker Pool, SimbaServer will create new Worker Threads and add them to the pool. When SimbaServer drops a TCP/IP connection, it places the Worker Thread back into the Worker Pool for assignment to a new connection.

## Worker Thread

SimbaServer assigns a Worker Thread to each TCP/IP connection for the duration of the connection. The Worker Thread waits on the task queue for tasks to be placed there by the Select Thread in response to requests from SimbaClient. When a task arrives, the worker removes the task from the queue and executes it by making method calls into the DSI implementation. If there is a response to the method calls, the Worker Thread sends it back to SimbaClient via the TCP/IP connection.

## 9.2 SimbaClient for ODBC Architecture

SimbaClient for ODBC is a multi-threaded DLL designed to deliver a responsive feel to the end user. Carefully designed threading and caching make sure that SimbaClient delivers the first data to the application as soon as possible. This allows the application to respond quickly to the end user's inputs so the end user can resume working.

This section explains how the threading model used in SimbaClient works and how it improves the response of the application for the user.

### Main Thread

The end user's application calls into SimbaClient on the Main Thread. This thread handles all requests sent from the application to SimbaServer including login, prepare, execute and disconnect. The main thread builds request messages and sends them through the Communications Manager to the server. The Main Thread reads non-result set responses from the Communications Manager and result set data from the memory cache

## Read Thread

The Main Thread creates the Read Thread, which then runs independently. The Read Thread reads all response messages received from SimbaServer regardless of type. It sends the response messages to the Communications Manager to be stored in a queue assigned to the appropriate TCP/IP connection. The Main Thread and the Fetch Thread remove the response messages from the queue and handle them as needed by the user's application.

## Communications Manager

The Communications Manager sends all request messages to the server and stores all response messages from the server. SimbaClient will receive at least one response message from SimbaServer for every message sent by the Communications Manager. In the case of result set data, a single request could result in a stream of multiple data responses.

## Fetch Thread

Immediately following execution of a SQL statement, the Fetch Thread sends a 'ReadData' request to SimbaServer. This begins the process of fetching the result set data from SimbaServer to SimbaClient and it starts regardless of whether the application has requested data yet. The Fetch Thread will continue to send 'ReadData' requests and fetch data until the cursor has reached the end of the result set. The data is stored in SimbaClient's data cache for later access by the application.

## Data Cache

The Fetch Thread populates the data cache with result set data. The Main Thread retrieves data from this cache as the application requests it. With this mechanism, if SimbaClient can retrieve data faster than the application requests it, the application will never be starved for data and the end user will never see the application wait for data.

## Appendix A: How Do I Get Support?

We welcome your questions and comments. To help us help you faster please have ready a detailed summary of your machine environment (operating system, version, patch-level, etc.) before you contact us. Providing us with this information helps us to understand your situation and to help you more effectively.

### Before contacting Customer Support:

To help us assist you more quickly, please have the following information ready when you contact us:

- The platform and operating system version for both the server and client computer(s)
- Your SimbaEngine SDK product version.
- The contents of the following files:

#### On Windows SimbaServers

File	Information
server	Registry key under HKEY_LOCAL_MACHINE\SOFTWARE\Simba\Server
driver	Registry key under HKEY_LOCAL_MACHINE\SOFTWARE\Simba\Driver
odbc	Registry key under HKEY_LOCAL_MACHINE\SOFTWARE\ODBC and/or HKEY_CURRENT_USER\SOFTWARE\ODBC
odbcinst	Registry key under HKEY_LOCAL_MACHINE\SOFTWARE\ODBC

#### On Windows SimbaClients

File	Information
driver	Registry key under HKEY_LOCAL_MACHINE\SOFTWARE\Simba\Driver
odbc	Registry key under HKEY_CURRENT_USER\SOFTWARE\ODBC
odbcinst	Registry key under HKEY_LOCAL_MACHINE\SOFTWARE\ODBC

#### On UNIX and Linux SimbaServers and SimbaClients

File	Information
.odbc.ini	File in SimbaServer user's home directory
.profile	File in SimbaServer user's home directory
.simba.ini	File in SimbaServer user's home directory

## How to Contact Us

### By telephone:

Customer Support: +1.604.633.0008 ext 3. Customer Support is available Monday to Friday, from 9 a.m. until 5 p.m. Pacific Time.

### By fax or e-mail:

Fax: +1.604.633.0004

Send e-mail to [support@simba.com](mailto:support@simba.com)

### On the Web:

Visit us on the Web at [www.simba.com](http://www.simba.com) and submit technical requests online at [www.simba.com/support.htm](http://www.simba.com/support.htm)