

Altibase® Tools & Utilities

# Altibase SSL/TLS User's Manual

Release 7.1 (September 18, 2017)



Altibase® Tools & Utilities Altibase SSL/TLS User's Manual  
Release 7.1  
Copyright © 2001~2017 Altibase Corp. All rights reserved.

This manual contains proprietary information of Altibase Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright patent and other intellectual property law. Reverse engineering of the software is prohibited.  
All trademarks, registered or otherwise, are the property of their respective owners.

Altibase Corp.  
10F, Daerung PostTower II,  
306, Digital-ro, Guro-gu, Seoul 08378, Korea  
Telephone: +82-2-2082-1000 Fax: 82-2-2082-1099  
Homepage: <http://www.altibase.com>

# Contents

<b>Preface</b> .....	<b>5</b>
About This Manual .....	6
Audience .....	6
Software Environment.....	6
Organization.....	6
Documentation Conventions.....	6
Related Documents.....	8
Online Manuals.....	8
Altibase Welcomes Your Opinions.....	8
<b>1. Introduction to Altibase SSL/TLS</b> .....	<b>9</b>
1.1 What is SSL/TLS .....	10
1.1.1 Introduction to SSL/TLS Protocols.....	10
1.1.2 Cryptography and Certificates .....	10
1.2 Secure Communication in Altibase.....	11
1.3 SSL Characteristics in Altibase.....	12
<b>2. Installing and Starting SSL in Altibase</b> .....	<b>13</b>
2.1 Software Requirements.....	14
2.1.1 Server .....	14
2.1.2 Client .....	14
2.2 Configuring the Environment for SSL.....	15
2.2.1 Configure SSL on the Server.....	15
2.2.2 Configure SSL for JDBC .....	17
2.2.3 Configure SSL for ODBC .....	22
<b>3. Managing SSL Connections</b> .....	<b>27</b>
3.1 Managing SSL Connections.....	28
3.1.1 Limit TCP Connections .....	28
3.1.2 Monitor and Manage SSL Connections.....	29
<b>Appendix A. SSL Sample</b> .....	<b>31</b>
Sample Using SSL Connection in JDBC.....	31



# Preface

---

# About This Manual

This manual discusses how to configure and use the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols within Altibase.

## Audience

This manual has been prepared for the following Altibase users:

- Database administrators
- Developers
- Technical support staff
- Security staff

It is recommended that those reading this manual possess the following background knowledge:

- Basic knowledge in the use of computers, operating systems, and operating system utilities
- Experience in using relational databases and an understanding of database concepts
- Experience in server management, operating systems or network management
- Basic knowledge of data storage, management and transactions in a distributed environment
- Basic knowledge of SSL/TLS protocols and secure communication

## Software Environment

This manual has been prepared assuming that Altibase 7.1 is used as the database server.

## Organization

This manual has been organized as follows:

- [Chapter1: Introduction to Altibase SSL/TLS](#)
- [Chapter2: Installing and Starting SSL in Altibase](#)
- [Chapter3: Managing SSL Connections](#)
- [Appendix A. SSL Sample](#)

## Documentation Conventions

This section describes the convention used in this manual. Understanding this convention will

make it easier to find information in this manual and other manuals in the series.

The following convention is explained:

- Sample code conventions

## Sample Code Conventions

The code examples explain SQL statements, stored procedures, iSQL statements, and other command line syntax.

The following table describes the printing conventions used in the code examples.

Rules	Meaning	Example
[ ]	Indicates an optional item.	VARCHAR [(size)] [[FIXED ]] VARIABLE]
{ }	Indicates a mandatory field for which one or more items must be selected.	{ ENABLE   DISABLE   COMPILE }
	A delimiter between optional or mandatory arguments.	{ ENABLE   DISABLE   COMPILE } [ ENABLE   DISABLE   COMPILE ]
.	Indicates that the previous argument is repeated, or that sample code has been omitted.	iSQL> select e_lastname from employees; E_LASTNAME ----- Moon Davenport Kobain . . . 20 rows selected.
Other Symbols	Symbols other than those shown above are part of the actual code.	EXEC :p1 := 1; acc NUMBER(11,2);
Italics	Statement elements in italics indicate variables and special values specified by the user.	SELECT * FROM table_name; CONNECT userID/password;
Lower case words	Indicate program elements set by the user, such as table names, column names, file names, etc.	SELECT e_lastname FROM employees;

Upper case words	Keywords and all elements provided by the system appear in upper case.	DESC SYSTEM_.SYS_INDEX_;
------------------	--	--------------------------

## Related Documents

For more detailed information, please refer to the following documents:

- Administrator's Manual
- Error Message Reference
- General Reference
- iSQL User's Manual
- SQL Reference

## Online Manuals

Online versions of our manuals (PDF or HTML) are available from Altibase's Customer Support site (<http://altibase.com/support-center/>).

## Altibase Welcomes Your Opinions

Please feel free to send us your comments and suggestions regarding this manual. Your comments and suggestions are important to us, and may be used to improve future versions of the manual. When you send your feedback, please make sure to include the following information:

- The name and version of the manual you are using
- Your comments and suggestions regarding the manual
- Your full name, address, and phone number

In addition to suggestions, this address may also be used to report any errors or omissions discovered in the manual, which we will address promptly.

If you need immediate assistance with technical issues, please contact Altibase's Customer Support site (<http://altibase.com/support-center/>).

We always appreciate your comments and suggestions.



# 1. Introduction to Altibase SSL/TLS

---

This chapter introduces Altibase SSL/TLS.

# 1.1 What is SSL/TLS

This chapter introduces the basic concepts of Altibase SSL/TLS.

## 1.1.1 Introduction to SSL/TLS Protocols

Secure Sockets Layer (SSL) is the de facto standard network security protocol originally developed by Netscape Communications Corporation.

SSL 1.0 was not publicly released. A bug-ridden 2.0 was released and followed by a completely redesigned 3.0 in 1996. Afterwards, the Internet Engineering Task Force (IETF) released TLS 1.0, which was an upgrade of 3.0 defined in RFC 2246. For further information about the TLS protocol, please refer to 'RFC-2246 The TLS Protocol (<ftp://ftp.rfc-editor.org/in-notes/rfc2246.txt>)'.

## 1.1.2 Cryptography and Certificates

Symmetric-key algorithms have two or more parties share a single secret key to ensure communication security. This secret key encrypts plaintext and decrypts ciphertext with an efficient algorithm. However, it is difficult for the communicating parties to safely exchange the secret key.

Asymmetric cryptography (also called public-key cryptography) was introduced thereupon. It uses a mathematically linked key pair of one public key and one private key. A message encrypted by a freely accessible public key can only be decrypted by the owner of the matching private key and vice versa.

One party encrypts the secret key with the counterparty's public key and sends it to the counterparty (the holder of the corresponding private key). The counterparty then decrypts the secret key with its own private key. Afterwards the communicating parties can safely communicate by encrypting and decrypting.

However, a man-in-the-middle (MITM) attacker can pretend to be both server and client. It is possible that a request for a public key is intercepted and the attacker's public key is sent, instead of the legitimate one.

Public key ownership can be verified with a public key certificate (a digital certificate signed by an official certificate authority (CA) trusted by both parties). In an opened-system environment such as the World Wide Web (WWW), it is important to use public certificates for communication security. On the contrary, a private certificate is a certificate generated and signed by a non-official CA. Private certificates are useful in a closed-system environment because they are free to use and can be generated whenever necessary.

## 1.2 Secure Communication in Altibase

Altibase adopts SSL/TLS using symmetric-key algorithms to encrypt/decrypt data, and asymmetric cryptography to safely exchange the shared key and public/private key pair for authentication.

When a network connection over SSL is initiated, Altibase and its client perform an SSL handshake as below:

1. The client and server exchange their information (e.g., SSL version, cipher setting and so on) to set up a secure connection environment.
2. If server authentication is required, the server sends the client its own certificate and the client verifies server identity with it.
3. If the server requests for the client's certificate, the client also sends the server its own certificate and the server verifies client identity with it.
4. The client and server exchange key information with public key cryptography and generate a session key to encrypt/decrypt data and validate data integrity.
5. The client and server communicate to each other that the future message will be encrypted with the session key and that the handshake is finished.

## 1.3 SSL Characteristics in Altibase

The following list of features are the SSL characteristics when using Altibase with SSL communication:

- Altibase uses the TLS 1.0 protocol supported by the OpenSSL library.
- Altibase supports server-only authentication and mutual authentication.
  - Server-only authentication: Only the server authenticates itself to the client.
  - Mutual authentication: Both server and client concurrently authenticate themselves to each other.
- An additional network port such as port 443 for HTTPS (other than the one that was used) is required to use SSL connections within Altibase. This is because Altibase does not allow a connection switch from non-secure to secure in TCP connections.
- To use a secure connection, all client applications need to be implemented with Java Secure Socket Extension (JSSE) API, which is integrated into Java 1.5 or above. JSSE provides a framework and implementation for a Java version of SSL 2.0, 3.0, and TLS 1.0 protocols as well as data encryption, server authentication, message integrity and optional client authentication.
- Altibase provides the JDBC and ODBC interfaces for SSL connection, which is currently supported only in Intel-Linux.

# 2. Installing and Starting SSL in Altibase

---

This chapter explains how to install SSL and required software.

## 2.1 Software Requirements

This section discusses the requirements necessary to use SSL on the server and the client.

### 2.1.1 Server

- OpenSSL toolkit 0.9.4~1.0.2
- Altibase version 6.5.1 or above (only supports Intel-Linux)

The OpenSSL toolkit is a prerequisite for using SSL/TLS in Altibase. The OpenSSL toolkit was developed by the OpenSSL Project and can be downloaded from <http://www.openssl.org/source>. You should verify that the installed OpenSSL version is not vulnerable to the Heartbleed bug that was disclosed in April 2014.

You can use the `OPENSSL_NO_HEARTBEATS` option to check whether or not it is infected.

### 2.1.2 Client

#### 2.1.2.1 ODBC

The OpenSSL toolkit has to be installed in order to use the SSL communication with ODBC.

#### 2.1.2.2 JDBC

It is recommended using Java Runtime Environment 1.6 (JRE1.6) or above to conveniently implement the client Java application through the SSL. JRE 1.6 or above((JRE 1.5 are also available for use) is recommended for the following reasons:

- Only JRE 1.6 or above supports importing the client's certificate into the keystore (Note that importing the certificate is not always required.).

## 2.2 Configuring the Environment for SSL

This section discusses how to configure the environment for Altibase SSL.

- [Configure SSL on the Server](#)
- [Configuring the Environment for SSL](#)
- [Configure SSL for ODBC](#)

### 2.2.1 Configure SSL on the Server

- [Step 1: Confirm the Installation of OpenSSL and its Library](#)
- [Step 2: Set Server Properties to Connect over SSL](#)
- [Step 3: Specify SSL Client Authentication](#)
- [Step 4: Set Server Certificate, Private Key, and Certificate Authority](#)
- [Step 5: Start the Server](#)

#### 2.2.1.1 Step 1: Confirm the Installation of OpenSSL and its Library

You are recommended to install the OpenSSL toolkit before installing SSL-enabled Altibase. Otherwise, if an Altibase function is used and OpenSSL is not installed, Altibase reports that it is unable to find the OpenSSL library.

Verify that OpenSSL is installed on the server and that it is not infected with the Heartbleed bug. If necessary, install it with the package manager provided by the operating system (e.g., RPM, Red Hat Linux) or download it from <http://www.openssl.org/source> and compile it manually.

Once installation is complete, examine the version of the installed OpenSSL as below.

```
$ openssl version  
OpenSSL 0.9.7e-fips-rhel5 01 Jul 2008
```

#### 2.2.1.2 Step 2: Set Server Properties to Connect over SSL

The following are properties that you need to set to connect over SSL within Altibase. These properties are located in the `$ALTIBASE_HOME/conf` file. For further information about these properties, please refer to the *General Reference*.

- `SSL_ENABLE`  
Switches the SSL feature on or off within Altibase. To enable, set the value to 1.
- `SSL_PORT_NO`

Specifies the listening port number for SSL connections. The value must be unique.

- **SSL\_MAX\_LISTEN**  
Sets the listen queue maximum size for concurrent SSL connections. You should note that more listeners require more memory.
- **SSL\_CIPHER\_LIST**  
The candidate cipher algorithm list available for the server and client to use and negotiate with. Depending on your security policy, you can specify one or more cipher names and separate them by colons. You can check the list at OpenSSL (<http://www.openssl.org/>) or execute the following command in the shell environment.

```
$ openssl ciphers:
```

### 2.2.1.3 Step 3: Specify SSL Client Authentication

- **SSL\_CLIENT\_AUTHENTICATION**  
Sets the SSL authentication mode to server-only authentication or mutual authentication. If you set the value to 1 for mutual authentication, the server will request a certificate from the client during the SSL handshake.

### 2.2.1.4 Step 4: Set Server Certificate, Private Key, and Certificate Authority

- [SSL\\_CERT](#)
- [SSL\\_KEY](#)
- [SSL\\_CA, SSL\\_CAPATH](#)
- 

#### **SSL\_CERT**

Sets the server certificate path.

For example, if the name of a server certificate is server-cert.pem and it is located in the \$ALTIBASE\_HOME/cert directory, the value for this property is ?/cert/server-cert.pem.

#### **SSL\_KEY**

Sets the server private (secret) key path.

For example, if the name of a server's secret key is server-key.pem and it is located in the \$ALTIBASE\_HOME/cert directory, the value for this property is ?/cert/server-key.pem.

#### **SSL\_CA, SSL\_CAPATH**

You need to specify SSL\_CA or SSL\_CAPATH to load CA certificate files to certify the ownership



of the received certificates. CA certificates can exist in a user-specific file path or a X.509 structured directory.

If you don't have an accredited certificate, you can use SSL by creating a private certificate.

### 2.2.1.5 Step 5: Start the Server

Start SSL-enabled Altibase. Please refer to the sample files provided in the Appendix.

If SSL\_ENABLE is set to 1, the SSL listen port is displayed as follows. This means that both the environment variables ALTIBASE\_PORT\_NO and ALTIBASE\_SSL\_PORT\_NO need to be defined.

```
$server start
-----
Altibase Client Query utility.
Release Version 7.1.0.0.1
Copyright 2000, Altibase Corporation or its subsidiaries.
All Rights Reserved.
-----
ISQL_CONNECTION = UNIX, SERVER = localhost
[ERR-910FB : Connected to idle instance]
Connecting to the DB server.... Connected.
...
TRANSITION TO PHASE : SERVICE
[CM] Listener started : TCP on port 20300 [IPV4]
[CM] Listener started : SSL on port 20443 [IPV4]
...
--- STARTUP Process SUCCESS ---
Command executed successfully.
```

## 2.2.2 Configure SSL for JDBC

- [Step 1: Confirm the Installation of OpenSSL and its Library](#)
- [Step 2: Set Server Properties to Connect over SSL](#)
- [Step 3: Specify SSL Client Authentication](#)
- [Step 4: Set Server Certificate, Private Key, and Certificate Authority](#)

### 2.2.2.1 Step 1: Import Certificates

The first step is to import certificates into the truststore for the server's CA certificate or into the keystore for its own CA certificate and secret key. Java Secure Socket Extension (JSSE) uses the truststore and keystore for authentication.

The next step to take depends on whether the CA certificate type is public or private and the job thereafter depends on whether the chosen authentication mode is server-only authentication or mutual authentication.

1. Private certificates with server-only authentication mode:  
Import the server's CA certificate into the truststore.
2. Private certificates with mutual authentication mode:  
Import the server's CA certificate into the truststore. Import the PKCS #12 formatted file, containing its own CA certificate and secret key, into the keystore.
3. Public certificates with server-only authentication mode:  
No action is necessary.
4. Public certificates with mutual authentication mode:  
Import the PKCS #12 formatted file, containing its own CA certificate and secret key, into the keystore.

When using private certificates (1 or 2 above), import the server's CA certificate into the truststore.

```
$keytool -import -alias alias_name -file server_certificate_file.pem -keystore truststore -storepass password
```

For 2 or 4, prior to importing its certificate and secret key, ensure that the PKCS #12 formatted file, containing its own certificate and secret key, is ready and that the Java version is 1.6 or above.

If the PKCS #12 file is not ready, execute OpenSSL with the pkcs12 option to generate the PKCS #12 file which contains the client's certificate and secret key as follows.

```
$openssl pkcs12 -export -in client_certificate.pem -inkey client_secretkey_file.pem > pkcs_file.p12
```

Import the prepared PKCS #12 file into the keystore with the following options. <sup>1</sup>

```
$keytool -importkeystore -srckeystore pkcs_file.p12 -destkeystore keystore.jks -srcstoretype pkcs12
```

### 2.2.2.2 Step 2: Set up Authentication in a Java Environment

JRE must access the truststore and keystore for authentication over SSL. The user can select from any of the following three configurations.

- [Configure a Java Application's Command-line Option](#)
- [Use the System.setProperty Method in a Java Application](#)
- [Configure the JDBC Property in a Java Application](#)
- 

#### Configure a Java Application's Command-line Option

```
-Djavax.net.ssl.keyStore=path_to_keystore  
-Djavax.net.ssl.keyStorePassword=password
```

---

<sup>1</sup>You can only import .pem files into keystore with the '-importkeystore' option in Java 6 or above.

```
-Djavax.net.ssl.trustStore=path_to_truststore
-Djavax.net.ssl.trustStorePassword=password
```

### Use the System.setProperty Method in a Java Application

```
System.setProperty("javax.net.ssl.keyStore", "path_to_keystore");
System.setProperty("javax.net.ssl.keyStorePassword", "password");
System.setProperty("javax.net.ssl.trustStore", "path_to_truststore");
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

### Configure the JDBC Property in a Java Application

```
Properties sProp = new Properties();
sProps.put("keystore_url", "path_to_keystore");
sProps.put("keystore_password", "password");
sProps.put("truststore_url", "path_to_truststore");
sProps.put("truststore_password", "password");
```

### 2.2.2.3 Step 3: Set JDBC Properties for SSL

Altibase provides JDBC for SSL connection to use SSL within the database. JDBC properties are categorized into the following two groups:

- [JDBC Properties for SSL Connections](#)
- [JDBC Properties for Authentication](#)
- 

**JDBC Properties for SSL Connections**

Name	Description	Range	Default Value
ssl_enable	Specifies whether or not to connect to the database over SSL connection. An SSL connection is created if this value is true; a TCP connection is created if this value is false.	[true   false ]	false
port	Specifies the SSL port number on the target server. The priority for an SSL port number is: 1) if ssn_enable is true and a value has been specified, this value is applied first. 2) if ssn_enable is true and this value is omitted, the ALTIBASE_SSL_PORT_NO environment variable is applied. 3) if ssn_enable is true and both this value and the ALTIBASE_SSL_PORT_NO environment variable are omitted, the	0 ~ 65535	ssl_enable(false): 20300 ssl_enable(ture): 20443

Name	Description	Range	Default Value
	default value (20300) is applied.		
ciphersuite_list	This is a list of available ciphers. Each name is separated by a colon (e.g., SSL_RSA_WITH_RC4_128_MD5:SSL_RSA_WITH_RC4_128_SHA). If JRE does not support the named algorithm, an IllegalArgumentException with the "Unsupported ciphersuite" message is thrown.	String	All cipher suite lists supported by JRE

#### JDBC Properties for Authentication

Name	Description	Range	Default Value
verify_server_certificate	Specifies whether or not to authenticate the target server's CA certificate. To use SSL exception authentication, set as false and the client application will not authenticate the server's CA certificate. In this case, it is unnecessary to import the server's private CA certificate for authentication.	[ true   false ]	true
keystore_url	Specifies the path to the keystore (a container for its own private key and the certificates with their corresponding public keys).	String	N/A

keystore_type	Specifies the keystore type for keystore_url. Java Key Store (JKS) is the most common type in Java.	[ JKS, JCEKS, PKCS12, etc.]	JKS
keystore_password	Specifies the password for keystore_url.	String	N/A
truststore_url	Specifies the path to the truststore (a keystore containing certificates that belong to the communication partners).	String	N/A
truststore_type	Specifies the truststore type at truststore_url above. Java Key Store (JKS) is the most common type in Java.	[JKS, JCEKS, PKCS12, etc.]	JKS
truststore_password	Specifies the password to truststore_url above.	String	N/A

#### 2.2.2.4 Step 4: Set Altibase Environment Variables

Altibase sets the port number for SSL communication. This step can be omitted.

If the port property has not been set for JDBC, the value specified for ALTIBASE\_SSL\_PORT\_NO will be used as the port number. If ALTIBASE\_SSL\_PORT\_NO is omitted, the default value for the port property is used.

Name	Description	Range	Default Value
ALTIBASE_SSL_PORT_NO	Specifies the SSL port number on the target server.	1024 ~ 65535	N/A

#### 2.2.2.5 SSL Issues in JDBC

Please consider the following when using SSL for JDBC.

##### Importing the PKCS #12 File into the Keystore (Available for JRE 1.6. or Above)

To use mutual authentication over SSL, you need to import the client's CA certificate and secret key into the keystore. This feature is supported for JRE 1.6 or above because you can only use the -importkeystore option to send a pem format file to the keystore in Java 6.0 or above.

However, as explained in Step 1, Java 6 or above can import pem-formatted files into the keystore with the `-importkeystore` option. Therefore, mutual authentication is not supported for JRE 1.5 or below.

Yet, once you have imported a PKCS file in JRE 1.6 or above, you can use JRE 1.5. In other words, JRE 1.6 is a requirement for importing only; mutual authentication can be performed in 1.5.

```
$keytool -importkeystore -srckeystore pkcs_file.p12 -destkeystore keystore.jks -srcstoretype pkcs12
```

## 2.2.3 Configure SSL for ODBC

- [Step 1: Check the OpenSSL Library](#)
- [Step 2: Prepare the Client's Certificate](#)
- [Step 3: Set ODBC/CLI Properties for SSL](#)
- [Step 4: Write a Client Program](#)

### 2.2.3.1 Step 1: Check the OpenSSL Library

You need the OpenSSL library to use SSL in ODBC. ODBC reads the OpenSSL library and calls the necessary functions while connecting to the server.

Therefore, you should verify that the OpenSSL library has been installed properly, before writing a client application. Where the library is installed can differ among operating systems.

- Check that the library has been installed:

```
$ ls -al /usr/lib/libssl*  
$ ls -al /usr/lib/libcrypto*
```

- Check that the utility has been installed:

```
$ openssl version
```

### 2.2.3.2 Step 2: Prepare the Client's Certificate

Prepare the client's certificate and secret key in a pem format file for mutual authentication of the server and client. The location of these files should be accessible by a client using ODBC.

### 2.2.3.3 Step 3: Set ODBC/CLI Properties for SSL

You need to set SSL properties appropriately, before writing a client program using SSL. The client can specify the following properties as a connection string when connecting to the server. Refer to the sample program for usage.

SSL connection properties are located in `$ALTIBASE_HOME/conf`.

Name	Description	Range	Default Value
SSL_CA	Specifies the file path to store CA certificates to certify the ownership of received certificates. CA certificates can exist in a user-specific file path or a X.509 structured directory. Ex)SSL_CA= /cert/ ca-cert.pem.	N/A	NULL
SSL_CAPATH	Specifies CAPATH in a CA directory format. Ex) SSL_CAPATH=/etc/ssl/certs	N/A	NULL
SSL_CERT	Sets the Altibase certificate path. Ex)SSL_CERT=/cert/client-cert.pem	N/A	NULL
SSL_KEY	Sets the server private (secret) key path. Ex) SSL_KEY=/cert/client-key.pem	N/A	NULL
SSL_VERIFY	Sets whether or not to authenticate the server certificate. An SSL handshake fails if authentication fails, and it becomes impossible to communicate over SSL. 0 : (OFF) Does not authenticate the server certificate 1: (ON) Authenticates the server certificate Ex) SSL_VERIFY=0	0: OFF 1: ON	0 (off)
SSL_CIPHER	A cipher algorithms available for the server and client to use and negotiate with. Depending on your security policy, you can specify one or more cipher names and separate them by colons(:). You can check the list at OpenSSL	N/A	NULL

	( <a href="http://www.openssl.org/">http://www.openssl.org/</a> ) or execute the following command in the shell environment. \$ openssl ciphers Ex)SSL_CIPHER=EDH-DSS-DES-CBC-SHA:DES-CBC-SHA		
--	---	--	--

The following table compares the SSL property of the server with the property for ODBC / CLI.

Name	Server (altibase.properties)	ODBC/CLI
SSL_ENABLE	O	X The client is same meaning with SSL_ENABLE = 1 if CONNTYPE = SSL
SSL_PORT_NO	O	X On the client, connect with CONNTYPE = SSL; PORT = 20443 without a separate SSL port.
SSL_MAX_LISTEN	O	X
SSL_CLIENT_AUTHENTICATION	O	X
SSL_CIPHER_LIST	O	X
SSL_CA	O	O
SSL_CAPATH	O	O
SSL_CERT	O	O
SSL_KEY	O	O



SSL_CIPHER	X	O
SSL_VERIFY	X	O

#### **2.2.3.4 Step 4: Write a Client Program**

Write a program to use SSL connection in the client application. You can find a sample program that uses SSL connection in the altibase directory. Please refer to `$ALTIBASE_HOME/sample/SQLCLI/SSL`.



# 3. Managing SSL Connections

---

## 3.1 Managing SSL Connections

This section discusses how to enhance security using the SSL exclusive connection feature, and monitor or manage SSL connections.

- [Limit TCP Connections](#)
- [Monitor and Manage SSL Connections](#)

### 3.1.1 Limit TCP Connections

The main reason to use SSL is to guarantee secure communication between the client and the server. In SSL-enabled Altibase, remote users are, by default, allowed to access the database over SSL, as well as TCP. Tightened security may require stricter access control of each database user.

The SYS user can control each user's access to the database only via SSL connection by disabling TCP connection as follows.

- Each user's TCP connection can be enabled or disabled with the ALTER USER statement:

```
ALTER USER user_name [ENABLE | DISABLE] TCP
```

- A new user can be created with TCP connection disabled using the CREATE USER statement:

```
CREATE USER user_name IDENTIFIED BY password [ENABLE | DISABLE] TCP
```

- Permission of TCP connection for database users can be checked by issuing the SELECT query on the DISABLE\_TCP column in the SYSTEM\_.SYS\_USERS\_ meta table.

```
SELECT * FROM SYSTEM_.SYS_USERS_
```

#### 3.1.1.1 Examples

<Query> Create a user who is not allowed to connect via TCP.

```
iSQL> CREATE USER user1 IDENTIFIED BY user1 DISABLE TCP;  
Create success.
```

<Query> Verify that the user is not allowed to connect via TCP.

```
iSQL> SELECT user_name, disable_tcp FROM SYSTEM_.SYS_USERS_;  
USER_NAME                                DISABLE_TCP  
-----  
SYSTEM_                                  F  
SYS                                      F  
USER1                                    T  
3 rows selected.
```

<Query> Grant a user who is not allowed to connect via TCP the permission to use TCP.

```
iSQL> ALTER USER user1 ENABLE TCP;
Alter success.
iSQL> SELECT user_name, disable_tcp FROM SYSTEM_.SYS_USERS_;
USER_NAME                                DISABLE_TCP
-----
SYSTEM_                                    F
SYS                                        F
USER1                                      F
3 rows selected.
```

### 3.1.2 Monitor and Manage SSL Connections

Monitoring active database connections is important to detect illegal access to the database. The V\$SESSION performance view provides detailed information on database connections such as the connected client, connection properties, and so on. In particular, the COMM\_NAME column describes the type of protocol, the client's address, and connected port number.

For further information about performance views, please refer to the *General Reference*.

```
iSQL> SELECT id, db_username, comm_name FROM V$SESSION WHERE comm_name like 'SSL%';
ID          DB_USERNAME
-----
COMM_NAME
-----
1          USER1
SSL 127.0.0.1:40328
1 row selected.
```

Once illegal access to the database is detected, the database administrator may need to forcefully terminate it. This can be done in the following steps.

- Log into the database in iSQL as the SYS user in SYSDBA mode and issue a query to disconnect the target session in iSQL.

```
$isql -s localhost -u user_name -p password -SYSDBA
iSQL> ALTER DATABASE database_name SESSION CLOSE session_number
```

- Ensure that the target session is disconnected by issuing a connection monitoring query.

```
iSQL> SELECT * FROM V$SESSION WHERE ID = session_id
```

SYSDBA is a special privilege for the SYS user to perform administrative jobs. Logging into the database with the SYS account in SYSDBA mode can be done in iSQL by inputting the -SYSDBA option when starting iSQL or using the CONNECT command with the AS SYSDBA option.

```
$ isql -s localhost -u sys -p manager -sysdba
-----
Altibase Client Query utility.
Release Version 7.1.0.0.1
Copyright 2000, Altibase Corporation or its subsidiaries.
All Rights Reserved.
-----
iSQL_CONNECTION = UNIX, SERVER = localhost
iSQL(sysdba)> SELECT id, db_username, comm_name FROM V$SESSION WHERE comm_name like 'SSL%';
ID          DB_USERNAME
-----
COMM_NAME
-----
```

```
1          USER1
SSL 127.0.0.1:40328
1 row selected.
iSQL(sysdba)> ALTER DATABASE mydb SESSION CLOSE 1;
Alter success.
iSQL(sysdba)> SELECT id, db_username, comm_name  FROM V$SESSION WHERE
comm_name like 'SSL%';
ID          DB_USERNAME
-----
COMM_NAME
-----
No rows selected.
```

# Appendix A. SSL Sample

---

Altibase provides a sample file in which the server and client use SSL connection.

Signed CA certificates, two pairs of certificates, secret key, sample java source code and more are available in `$ALTIBASE_HOME/sample/cert`.

You can find a sample program that uses SSL in the altibase directory in `$ALTIBASE_HOME/sample/`.

## Sample Using SSL Connection in JDBC

For a CLI sample program, please refer to `$ALTIBASE_HOME/sample/SQLCLI/SSL`.

```
import java.util.Properties;
import java.sql.*;

import Altibase.jdbc.driver.util.RuntimeEnvironmentVariables;

class SslSimpleSQL
{
    public static void main(String args[])
    {
        Properties      sProps    = new Properties();
        Connection      sCon      = null;
        Statement       sStmt     = null;
        PreparedStatement sPreStmt = null;
        ResultSet       sRS;

        if ( args.length == 0 )
        {
            System.err.println("Usage : java class_name port_no");
            System.exit(-1);
        }

        String      sDevhome = RuntimeEnvironmentVariables.getVariable("ALTIBASE_HOME");
        String      sTrusStore = sDevhome + "/sample/CERT/truststore";
        String      sKeyStore = sDevhome + "/sample/CERT/keystore.jks";

        // Keystore
        System.setProperty("javax.net.ssl.keyStore", sKeyStore);
        System.setProperty("javax.net.ssl.keyStorePassword", "altibase");

        // Truststore
        System.setProperty("javax.net.ssl.trustStore", sTrusStore );
        System.setProperty("javax.net.ssl.trustStorePassword", "altibase");

        String sPort      = args[0];
        String sURL       = "jdbc:Altibase://127.0.0.1/mydb";
        String sUser      = "SYS";
        String sPassword = "MANAGER";

        sProps.put( "user",      sUser);
        sProps.put( "password",  sPassword);
        sProps.put( "ssl_enable", "true");
        sProps.put( "ssl_port",  sPort);
    }
}
```

```

// sProps.put( "encoding", sEncoding );

/* Deploy Altibase's JDBC Driver */
try
{
Class.forName("Altibase.jdbc.driver.AltibaseDriver");
}
catch ( Exception e )
{
System.out.println("Can't register Altibase Driver");
System.out.println( "ERROR MESSAGE : " + e.getMessage() );
System.exit(-1);
}

/* Initialize environment */
try
{
sCon = DriverManager.getConnection( sURL, sProps );
sStmt = sCon.createStatement();
}
catch ( Exception e )
{
System.out.println( "ERROR MESSAGE : " + e.getMessage() );
e.printStackTrace();
}

try
{
sStmt.execute( "DROP TABLE TEST_EMP_TBL" );
}
catch ( SQLException e )
{
}

try
{
sStmt.execute( "CREATE TABLE TEST_EMP_TBL " +
"( EMP_FIRST VARCHAR(20), " +
"EMP_LAST VARCHAR(20), " +
"EMP_NO INTEGER )" );

sPreStmt = sCon.prepareStatement( "INSERT INTO TEST_EMP_TBL " +
"VALUES( ?, ?, ? )" );

sPreStmt.setString( 1, "Susan" );
sPreStmt.setString( 2, "Davenport" );
sPreStmt.setInt( 3, 2 );
sPreStmt.execute();

sPreStmt.setString( 1, "Ken" );
sPreStmt.setString( 2, "Kobain" );
sPreStmt.setInt( 3, 3 );
sPreStmt.execute();

sPreStmt.setString( 1, "Aaron" );
sPreStmt.setString( 2, "Foster" );
sPreStmt.setInt( 3, 4 );
sPreStmt.execute();

sPreStmt.setString( 1, "Farhad" );
sPreStmt.setString( 2, "Ghorbani" );
sPreStmt.setInt( 3, 5 );
sPreStmt.execute();

sPreStmt.setString( 1, "Ryu" );
sPreStmt.setString( 2, "Momoi" );

```



```

sPreStmt.setInt( 3, 6 );
sPreStmt.execute();

sRS = sStmt.executeQuery( "SELECT EMP_FIRST, EMP_LAST," +
" EMP_NO FROM TEST_EMP_TBL " );

/* Fetch all data */
while( sRS.next() )
{
System.out.println( " EmpName : " + sRS.getString(1) +
" " + sRS.getString(2) );
System.out.println( " EmpNO : " + sRS.getInt(3) );
}

/* Finalize process */
sStmt.close();
sPreStmt.close();
sCon.close();
}
catch ( SQLException e )
{
System.out.println( "ERROR CODE : " + e.getErrorCode() );
System.out.println( "ERROR MESSAGE : " + e.getMessage() );
e.printStackTrace();
}
}
}

```



# Index

<b>A</b>		<b>P</b>	
ALTIBASE_SSL_PORT_NO .....	21	port .....	19
Asymmetric cryptography .....	10	private certificate .....	10
		public key certificate .....	10
<b>C</b>		<b>S</b>	
ciphersuite_list .....	20	Secure Sockets Layer (SSL) .....	10
		Server-only authentication .....	12
<b>E</b>		SSL_CA .....	16
Endian function .....	31	SSL_CAPATH .....	16
		SSL_CERT .....	16
<b>J</b>		SSL_CIPHER_LIST .....	16
Java Secure Socket Extension (JSSE) API .....	12	SSL_CLIENT_AUTHENTICATION .....	16
		ssl_enable .....	19
<b>K</b>		SSL_ENABLE .....	15
keystore_password .....	21	SSL_KEY .....	16
keystore_type .....	21	SSL_MAX_LISTEN .....	16
keystore_url .....	20	SSL_PORT_NO .....	15
		Symmetric-key algorithms .....	10
<b>M</b>		<b>T</b>	
man-in-the-middle (MITM) .....	10	truststore_password .....	21
Mutual authentication .....	12	truststore_type .....	21
		truststore_url .....	21
<b>O</b>		<b>V</b>	
OpenSSL toolkit .....	14	verify_server_certificate .....	20
OPENSSL_NO_HEARTBEATS .....	14		