



## Table of Contents

<b>CONNECTION SETTINGS</b>	<b>2</b>
GENERAL SETTINGS FOR MYSQL	5
GENERAL SETTINGS FOR ORACLE	6
<i>Basic Connection General Settings</i>	7
<i>TNS Connection General Settings</i>	8
GENERAL SETTINGS FOR POSTGRESQL	9
GENERAL SETTINGS FOR SQLITE	10
SSH SETTINGS (AVAILABLE ONLY FOR MYSQL, ORACLE AND POSTGRESQL SERVERS AND SUPPORTS SSH2 PROTOCOL ONLY)	11
<i>Benefit of SSH Tunneling</i>	12
<i>Password Authentication</i>	13
<i>Public Key Authentication</i>	15
HTTP SETTINGS (AVAILABLE ONLY FOR MYSQL, POSTGRESQL AND SQLITE SERVERS)	17
<i>Uploading the Tunneling Script</i>	18
<i>Setting up HTTP Tunnel</i>	21
SSL SETTINGS (AVAILABLE ONLY FOR MYSQL AND POSTGRESQL SERVERS)	22
<i>Installation of OpenSSL and MySQL/PostgreSQL</i>	23
<i>Setting up SSL Certificate for MySQL/PostgreSQL</i>	24
<i>Setting up Client Certificate for Navicat</i>	27
ADVANCED SETTINGS	29
<i>Setting Advanced Database Properties (Available only for MySQL and PostgreSQL Servers)</i>	33

## Connection Settings

Navicat assembles utilitarian tools to manage your databases. To start managing your databases in Navicat, the first thing you require to do is to establish your Server connection.

### Create Connection

Navicat provides three typical approaches to establish your connection, click  or choose File ->  **New Connection** to start the setup.

- [General Settings for MySQL](#)
- [General Settings for Oracle](#)
- [General Settings for PostgreSQL](#)
- [General Settings for SQLite](#)
- [SSH Settings](#) (Available only for MySQL, Oracle and PostgreSQL Servers)
- [HTTP Settings](#) (Available only for MySQL, PostgreSQL and SQLite Servers)

**Note:** For MySQL or PostgreSQL server, a commonly-used protocol - **Secure Sockets Layer (SSL)** is employed for managing the security of a message transmission on the Internet (see [SSL Settings](#) for details).

Navicat provides evaluated accounts for testing purpose.

The remote MySQL server connection settings are:


- Host Name/IP Address: server1.navicat.com
- Port: 4406
- User Name: navicat
- Password: testnavicat

The remote PostgreSQL server connection settings are:

- Host Name/IP Address: server1.navicat.com
- Port: 5432
- Initial Database: template1
- User Name: navicat
- Password: testnavicat


**Note:** Navicat authorizes you to make connection to remote server running on different platform, i.e. Windows, Mac, Linux and UNIX.

To create a new connection with the same properties as one of the existing connection has

- Right-click the connection in the navigation pane and choose  **Duplicate Connection**.
- The newly created connection will be named as "connectionname\_copy".

## Delete Connection

To delete a connection

- Right-click the connection in the navigation pane and choose  **Delete Connection**.
- Confirm deleting in the dialog window.


## Open Connection

To open a connection

- Double-click the connection to open in the navigation pane.


## Close Connection

To close a connection

- Right-click the connection in the navigation pane and choose  **Close Connection**.

## Edit Connection

To edit a connection information

- Close the connection if it is being opened.
- Right-click the connection and choose  **Connection Properties**.

## Open Connection Settings Save Path

To open a connection settings save path

- Select the connection in the navigation pane.
- Right-click the connection and choose **Go to settings save path** or press **Ctrl+G** to open the settings save path folder.

## Export Connection Settings

To export connection settings

- Choose File -> **Export Connections**.
- Select the connections and the export file path.

## Import Connection Settings

To import connection settings

- Choose File -> **Import Connections**.
- Specify the connection settings file path.
- Confirm replacing or skipping in the dialog window if the connection already exists.

## Achieve Connection Information

To achieve a connection information

- Open the connection in the navigation pane.
- Right-click the opened connection and choose **Connection Information**.

## General Settings for MySQL

The following instruction guides you through the process of creating a new connection. To successfully establish a new connection to local/ remote MySQL Server - no matter via SSL, SSH or HTTP, set the connection properties in the corresponding boxes: Connection name, Host name, Port number, User name, and Password.

### Connection Name

A friendly name to best describe your connection.

### Host Name/IP Address

A host name where the database is situated or the IP address of the server.

### Port

A TCP/IP port for connecting to the database server.

### User Name

User name for connecting to the database server.

### Password

Password for connecting to the server.

See also:

[Advanced Settings](#)

Related topics:

[SSL](#), [SSH](#), [HTTP](#)

## General Settings for Oracle

The following instruction guides you through the process of creating a new connection for server. To successfully establish a new connection to local/remote Oracle Server - no matter via SSH, set the connection properties in the corresponding boxes: Connection name, Host name, Port number, User name, and Password.

Navicat supports 2 types of Oracle Server connection:

- [Basic Connection](#)
- [TNS Connection](#)

Related topics:

[SSH](#)

## Basic Connection General Settings

### Connection Name

A friendly name to best describe your connection.

### Connection Type

Connection type for connecting to the server: **Basic** or [TNS](#).

#### Basic

In Basic mode, Navicat connects to Oracle through the Oracle Call Interface (OCI). OCI is an application programming interface that allows an application developer to use a third-generation language's native procedure or function calls to access the Oracle database server and control all phases of SQL statement execution. OCI is a library of standard database access and retrieval functions in the form of a dynamic-link library.

### Host Name/IP Address

A host name where the database is situated or the IP address of the server.

### Port

A TCP/IP port for connecting to the database server.

### Service Name/SID

Set the Service Name/SID which the user connects when making connection. Select the corresponding radio button.

### User Name

User name for connecting to the database server.

### Password

Password for connecting to the server.

See also:

[Advanced Settings](#)

Related topics:

[TNS](#), [SSH](#)

## TNS Connection General Settings

### Connection Name

A friendly name to best describe your connection.

### Connection Type

Connection type for connecting to the server: [Basic](#) or **TNS**.

#### TNS

In TNS mode, Navicat connects to Oracle server using an alias entry from a tnsnames.ora file through the Oracle Call Interface (OCI). OCI is an application programming interface that allows an application developer to use a third-generation language's native procedure or function calls to access the Oracle database server and control all phases of SQL statement execution. OCI is a library of standard database access and retrieval functions in the form of a dynamic-link library.

### Net Service Name

The net service name.

### User Name

User name for connecting to the database server.

### Password

Password for connecting to the server.

See also:

[Advanced Settings](#)

Related topics:

[Basic](#), [SSH](#)

## General Settings for PostgreSQL

The following instruction guides you through the process of creating a new connection. To successfully establish a new connection to local/remote PostgreSQL Server - no matter via SSH, HTTP or SSL, set the connection properties in the corresponding boxes: Connection name, Host name, Port number, Initial Database, User name, and Password.

### Connection Name

A friendly name to best describe your connection.

### Host Name/IP Address

A host name where the database is situated or the IP address of the server.

### Port

A TCP/IP port for connecting to the database server.

### Initial Database

The initial database to which user connects when making connection.

### User Name

User name for connecting to the database server.

### Password

Password for connecting to the server.

See also:

[Advanced Settings](#)

Related topics:

[SSH](#), [HTTP](#)

## General Settings for SQLite

The following instruction guides you through the process of creating a new connection. To successfully establish a new connection to local/remote SQLite Server - no matter via HTTP, set the connection properties in the corresponding boxes: Connection name, Type and Database Name.

### Connection Name

A friendly name to best describe your connection.

### Type

Specify the type of database.

#### Existing Database File

Connect an existing database in the **Database File**.

#### New SQLite 3

Create a new SQLite 3 database in the **Database File**.

#### New SQLite 2

Create a new SQLite 2 database in the **Database File**.

### Database File

Specify the initial database file. If the [HTTP Tunnel](#) is enabled, you need to enter an absolute file path of the database file in your webserver.

See also:

[Advanced Settings](#)

Related topics:

[HTTP](#)

## SSH Settings (Available only for MySQL, Oracle and PostgreSQL Servers and supports SSH2 Protocol only)

**Secure SHell (SSH)** is a program to log in into another computer over a network, execute commands on a remote server, and move files from one machine to another. It provides strong authentication and secure encrypted communications between two hosts, known as **SSH Port Forwarding (Tunneling)**, over an insecure network. Typically, it is employed as an encrypted version of Telnet.

In a Telnet session, all communications, including username and password, are transmitted in plain-text, allowing anyone to listen-in on your session and steal passwords and other information. Such sessions are also susceptible to session hijacking, where a malicious user takes over your session once you have authenticated. SSH serves to prevent such vulnerabilities and allows you to access a remote server's shell without compromising security.

- [Benefit of SSH Tunneling.](#)

To ensure that the incoming connection request is from you, SSH can use a password, or public/private key pair (also called public key) authentication mechanism.

- [Password Authentication.](#)
- [Public Key Authentication.](#)

**Note:** Please make sure that the parameter - "AllowTcpForwarding" in the Linux Server must be set to value "yes", otherwise, the SSH port forwarding will be disabled. To look for the path: `/etc/ssh/sshd_config` .By default, the SSH port forwarding should be enabled. Please double check the value settings.

**\*\*** Even the server support SSH tunnel, however, if the port forwarding being disabled, Navicat cannot connect via SSH Port 22.

See also:

[Advanced Settings](#)

Related topic:

[SSL](#)

## Benefit of SSH Tunneling

SSH has a wonderful feature called SSH Port Forwarding, sometimes called SSH Tunneling, which allows you to establish a secure SSH session and then tunnel arbitrary TCP connections through it. Tunnels can be created at any time, with almost no effort and no programming, which makes them very appealing. SSH Port Forwarding can be used for secure communications in a myriad of different ways.

Many Hosting Companies that provide server hosting will block access to the Server from outside the hosting company's network, and only grant access to users connecting from localhost.

There are several benefits to using SSH:

- Connection to a server from behind a firewall when the server port is blocked.
- Automatic authentication of users, no passwords sent in plain text to prevent the stealing of passwords.
- Multiple strong authentication methods that prevent such security threats as spoofing identity.
- Encryption and compression of data for security and speed.
- Secure file transfer.



Related topics:

[Password Authentication](#), [Public Key Authentication](#)

## Password Authentication

Using this mode, SSH is almost identical to the program telnet. When you make a connection, you are asked for your password. You type it in and you are either logged in or denied. Your password is first encrypted and then sent over the network and then decrypted at the remote host. This is the mode that most users will be encouraged to use, as it requires no additional setup or configuration.

The following instruction guides you through the process of configuring a SSH connection using Password Authentication. To successfully establish a SSH connection, set the SSH connection properties in the corresponding boxes: Host name/IP address, Port number, User name, Authentication Method and Password.

1. Click  or choose File ->  **New Connection** to set up the Connection Properties.
2. Select the SSH tab and enable **Use SSH Tunnel**.
3. Fill in the required information:

### Host Name/IP Address

A host where SSH server is activated.

### Port

A port where SSH server is activated, by default it is 22.

### User Name

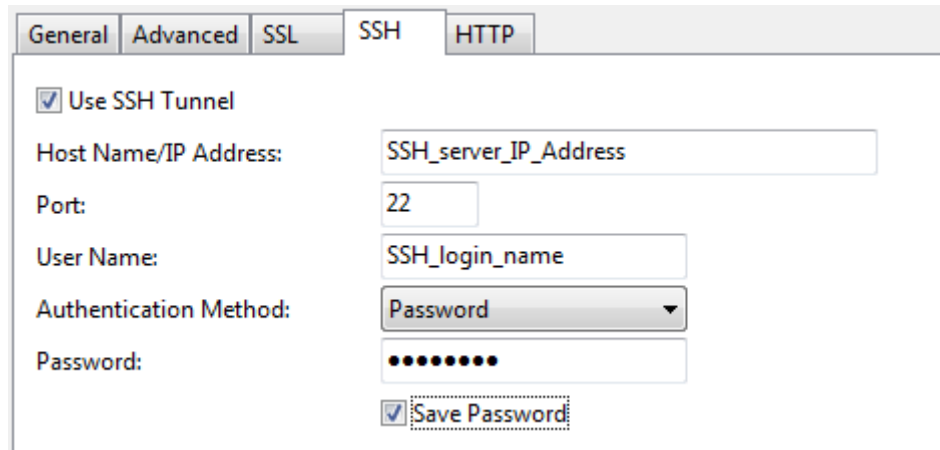
A user on Linux machine. (It is a Linux user. It is not a user of Database Server.)

### Authentication Method

Choose between Password Authentication and [Public Key Authentication](#)

### Password

It is a Linux user password.



The screenshot shows the SSH settings tab in Navicat. It includes a checked checkbox for 'Use SSH Tunnel', a text field for 'Host Name/IP Address' containing 'SSH\_server\_IP\_Address', a text field for 'Port' containing '22', a text field for 'User Name' containing 'SSH\_login\_name', a dropdown menu for 'Authentication Method' set to 'Password', a password field with masked characters, and a checked checkbox for 'Save Password'.

4. Navicat host name at the General Settings page ([MySQL](#), [Oracle](#) or [PostgreSQL](#)) should be set relatively to the SSH server which provided by your database hosting company.

See also:

[Advanced Settings](#)

Related topics:

[Public Key Authentication](#)



## Public Key Authentication

Public-key Authentication is based on the use of digital signatures and provides the best authentication security.

For Public Key Authentication to work four things are needed:

- the remote server(s) you are connecting must have your public key.
- the local client you are connecting from must have your private key.
- the remote server must be configured to allow you to login using your public key.
- the local client must be configured to use your private key while logging into remote server.

The following instruction guides you through the process of configuring a SSH connection using Public Key Authentication. To successfully establish a SSH connection , set the SSH connection properties in the corresponding boxes: Host name/IP address, Port number, User name, Authentication Method, Private Key and Passphrase.

1. Click  or choose File ->  **New Connection** to set up the Connection Properties.
2. Select the SSH tab and enable **Use SSH Tunnel**.
3. Fill in the required information:

### Host Name/IP Address

A host where SSH server is activated.

### Port

A port where SSH server is activated, by default it is 22.

### User Name

A user on Linux machine. (It is a Linux user. It is not a user of Database Server.)

### Authentication Method

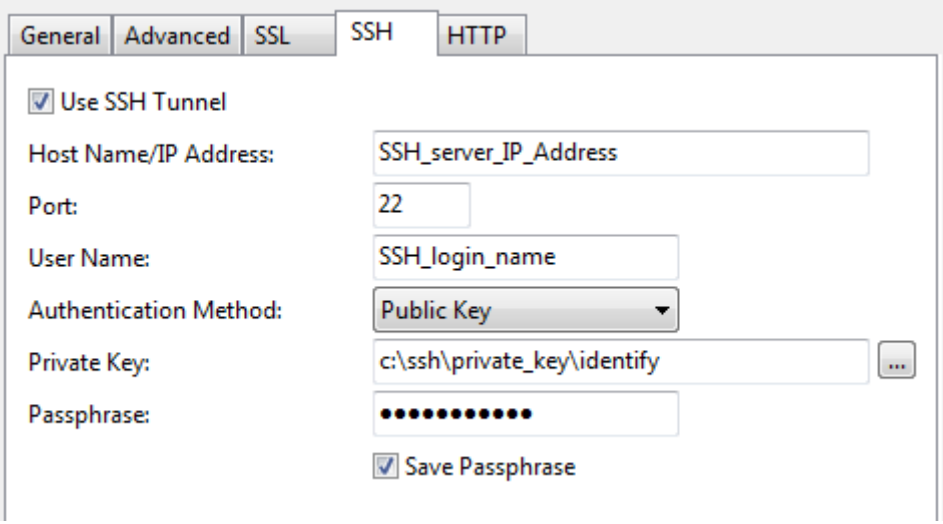
Choose between [Password Authentication](#) and Public Key Authentication

### Private Key

It is used together with your public key. The private key should be readable only by you.

## Passphrase

A passphrase is exactly like a password, except that it applies to the keys you are generating and not an account. The passphrase be any length under 1024 characters.



The screenshot shows the SSH settings dialog box in Navicat. The 'SSH' tab is selected. The 'Use SSH Tunnel' checkbox is checked. The 'Host Name/IP Address' field contains 'SSH\_server\_IP\_Address'. The 'Port' field contains '22'. The 'User Name' field contains 'SSH\_login\_name'. The 'Authentication Method' dropdown is set to 'Public Key'. The 'Private Key' field contains 'c:\ssh\private\_key\identify' and has a browse button. The 'Passphrase' field is filled with ten dots. The 'Save Passphrase' checkbox is checked.

4. Navicat host name at the General Settings page ([MySQL](#), [Oracle](#) or [PostgreSQL](#)) should be set relatively to the SSH server which provided by your database hosting company.

See also:

[Advanced Settings](#)

Related topics:

[Password Authentication](#)

## HTTP Settings (Available only for MySQL, PostgreSQL and SQLite Servers)

HTTP Tunneling is a method for connecting to a MySQL/PostgreSQL/SQLite server that uses the same protocol (http://) and the same port (port 80) as a webserver does. It is used while your ISPs do not allow direct connections to their MySQL/PostgreSQL server, but allows establishing HTTP connections.

Steps of setting up HTTP Connection for MySQL/PostgreSQL/SQLite Server:

1. [Uploading the Tunneling Script.](#)
2. [Setting up HTTP Tunnel.](#)

**Note:** HTTP Tunnel and SSH Tunnel cannot function simultaneously. The SSH Tunnel is disabled when you select the HTTP Tunnel and vice versa.

See also:

[Advanced Settings](#)

Related topics:

[General Settings for MySQL](#)

[General Settings for PostgreSQL](#)

[General Settings for SQLite](#)

## Uploading the Tunneling Script

To use this connection method, first thing you need to do is to upload the tunneling script - **ntunnel\_mysql.php**, **ntunnel\_pgsql.php** or **ntunnel\_sqlite.php** to the webserver where MySQL, PostgreSQL or SQLite server is located.

**Note:** **ntunnel\_mysql.php**, **ntunnel\_pgsql.php** or **ntunnel\_sqlite.php** is available in the Navicat installation folder.

The following instruction guides you through the process of uploading **ntunnel\_mysql.php** to your webserver. In this tutorial, you require a FTP client - [WS\\_FTP](#) to make connection to your webserver. You can use any FTP client you are familiar with.

1. Startup WS\_FTP and enter the connection information in the Quick Connection Bar.

### Address

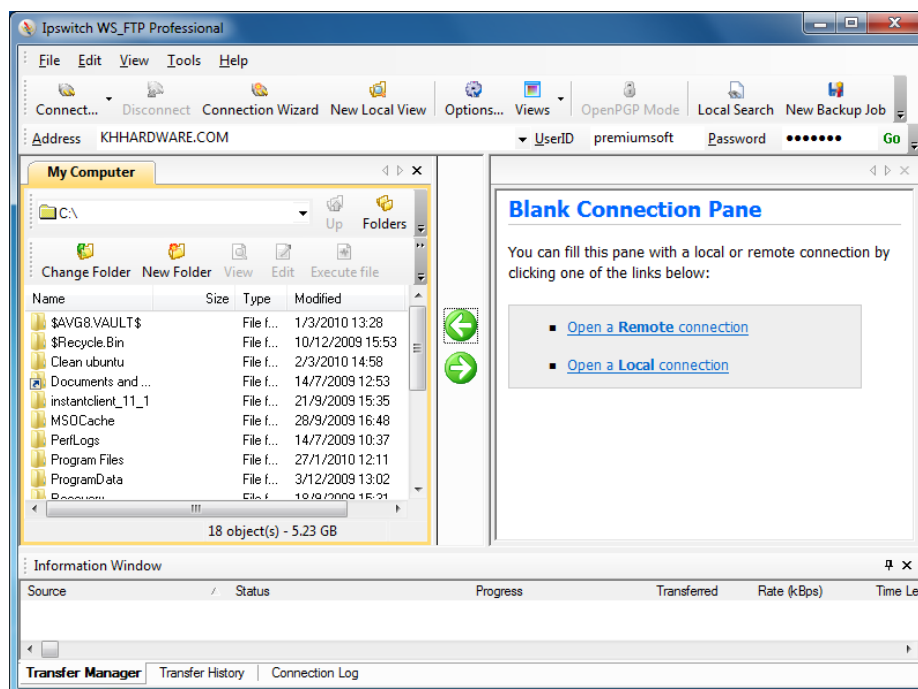
A fully qualified Internet host name or an IP address.

### User ID

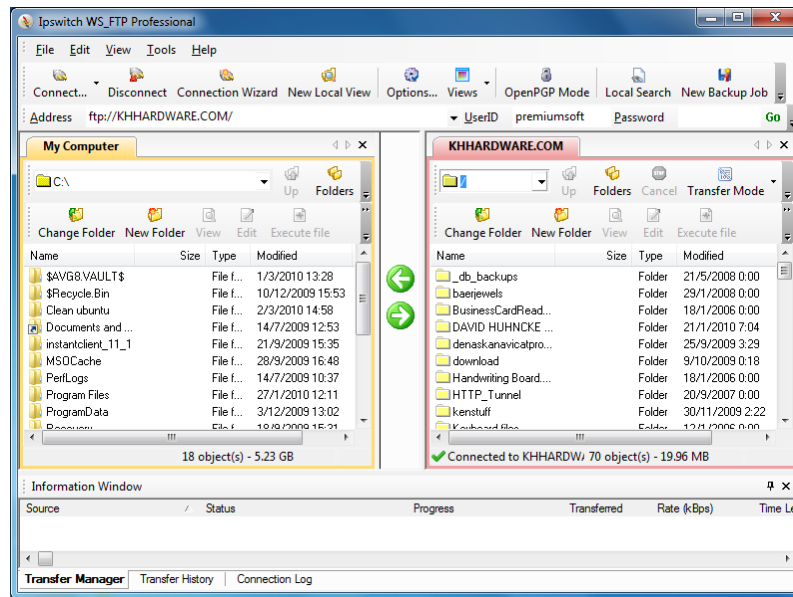
Enter the User ID you want to use for this connection.

### Password

Enter the Password you want to use for the User ID you entered.

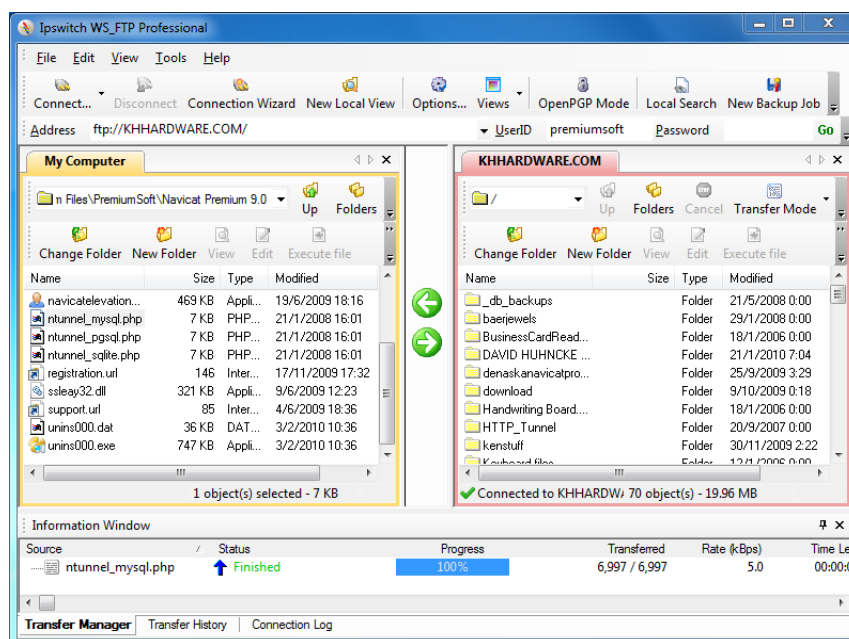


2. Click **Go** button to connect the webserver.



3. To upload the tunneling script:

1. Browse the `ntunnel_mysql.php` on the source system.
2. Open the directory to which you want to transfer files on the destination system.
3. Transfer the files using the left and right arrow buttons located between the list boxes, i.e. Click the right arrow button to transfer files from the local to the remote system.





See also:

[Advanced Settings](#)

## Setting up HTTP Tunnel

The following instruction guides you through the process of configuring a HTTP connection.

1. Click  or choose File ->  **New Connection** to set up the Connection Properties.
2. Select the HTTP tab and enable **Use HTTP Tunnel**.
3. Enter URL of the tunneling script, e.g. *http://www.navicat.com/ntunnel\_mysql.php* .
4. If the **ntunnel\_mysql.php**, **ntunnel\_pgsql.php** or **ntunnel\_sqlite.php** is hosted in a password protected server or you have to access internet over a proxy server, you can provide the required authentication details.
5. Navicat host name at the General Settings page ([MySQL](#), [PostgreSQL](#) or [SQLite](#)) should be set relatively to the HTTP server which provided by your database hosting company.

See also:

[Advanced Settings](#)

## SSL Settings (Available only for MySQL and PostgreSQL Servers)

Secure Sockets Layer(SSL) is a protocol for transmitting private documents via the Internet. To get a secure connection to work with MySQL/PostgreSQL Server, the first thing you need to do is to install OpenSSL Library and download MySQL/PostgreSQL Database Source.

Steps of setting up SSL Connection for MySQL/PostgreSQL Server and Navicat:

1. [Installation of OpenSSL and MySQL/PostgreSQL.](#)
2. [Setting up SSL Certificate for MySQL/PostgreSQL.](#)
3. [Setting up Client Certificate for Navicat.](#)

**Note:** Support from PostgreSQL 8.4 or later.

See also:

[Advanced Settings](#)

Related topics:

[General Settings for MySQL](#), [General Settings for PostgreSQL](#), [SSH](#)

## Installation of OpenSSL and MySQL/PostgreSQL

### Installing OpenSSL

1. Download OpenSSL - <http://www.openssl.org>
2. Linux command : [zcat 0.96l.tar.gz | tar xvf -]
3. Linux command : [./config]
4. Linux command : [make]
5. Linux command : [make install]

### Installing MySQL

1. Download MySQL - <http://www.mysql.com>
2. Linux command : [./configure --with -vio --with -openssl]
3. Linux command : [make]
4. Linux command : [make install]

**Note:** Please ensure if MySQL Server supports OpenSSL using query statement:  
[SHOW VARIABLES LIKE 'have\_openssl']; - Returns value = YES

### Installing PostgreSQL

1. Download PostgreSQL - <http://www.postgresql.org>
2. Linux command : [./configure --with-openssl]
3. Linux command : [gmake]
4. Linux command : [gmake install]

**Note:** Please ensure if PostgreSQL Server supports OpenSSL using query statement:  
[SHOW ssl;] - Returns value = ON

See also:

Step 2: [Setting up SSL Certificate for MySQL/PostgreSQL](#)

## Setting up SSL Certificate for MySQL/PostgreSQL

To create server/client side Certificate, login to the Linux Server as root and employ the Shell Command below:

### MySQL

1. `DIR=`pwd` /openssl`
2. `PRIV=$DIR/private`
3. `mkdir $DIR $PRIV $DIR/newcerts`
4. `cp /usr/share/ssl/openssl.cnf $DIR`
5. `replace ./demoCA $DIR -- $DIR/openssl.cnf`
6. Generation of Certificate Authority(CA)

```
/usr/local/ssl/bin/openssl req -new -x509 -keyout $PRIV/cakey.pem -out $DIR/cacert.pem -config $DIR/openssl.cnf
```

**Note:** If "PEM" is required, please enter different "PEM pass" via steps below.

7. Create server request and key

```
/usr/local/ssl/bin/openssl req -new -keyout $DIR/server-key.pem -out $DIR/server-req.pem -days 3600 -config $DIR/openssl.cnf
```

8. Remove the passphrase from the key (optional)

```
/usr/local/ssl/bin/openssl rsa -in $DIR/server-key.pem -out $DIR/server-key.pem
```

9. Sign server cert

```
/usr/local/ssl/bin/openssl ca -policy policy_anything -out $DIR/server-cert.pem -config $DIR/openssl.cnf -infiles $DIR/server-req.pem
```

10. Create client request and key

```
/usr/local/ssl/bin/openssl req -new -keyout $DIR/client-key.pem -out $DIR/client-req.pem -days 3600 -config $DIR/openssl.cnf
```

11. Remove a passphrase from the key (optional)

```
/usr/local/ssl/bin/openssl rsa -in $DIR/client-key.pem -out  
$DIR/client-key.pem
```

12. Sign client cert

```
/usr/local/ssl/bin/openssl ca -policy policy_anything -out $DIR/client-cert.pem  
-config $DIR/openssl.cnf -infiles $DIR/client-req.pem
```

13. Create a **my.cnf** file for testing the Certificates. Store it either in **/etc** or MySQL data directory (typically **/usr/local/var** for source installation)

**my.cnf** example content:

```
[client]  
ssl-ca=$DIR/cacert.pem  
ssl-cert=$DIR/client-cert.pem  
ssl-key=$DIR/client-key.pem  
[mysqld]  
ssl-ca=$DIR/cacert.pem  
ssl-cert=$DIR/server-cert.pem  
ssl-key=$DIR/server-key.pem
```

14. To start MySQL daemon:

```
/usr/local/libexec/mysqld -u mysql &
```

or

```
/usr/local/sbin/mysqld -u &
```

## PostgreSQL

1. To create a quick self-signed certificate for the server, use the following OpenSSL command:

```
openssl req -new -text -out server.reqm
```

2. Fill out the information that openssl asks for. Make sure you enter the local host name as "Common Name"; the challenge password can be left blank. The program will generate a key that is passphrase protected; it will not accept a passphrase that is less than four characters long. To remove the passphrase (as you must if you want automatic start-up of the server), run the commands:

```
openssl rsa -in privkey.pem -out server.key  
rm privkey.pem
```

3. Enter the old passphrase to unlock the existing key. Now do:

```
openssl req -x509 -in server.req -text -key server.key -out server.crt
```

4. to turn the certificate into a self-signed certificate and to copy the key and certificate to where the server will look for them. Finally do:

```
chmod og-rwx server.key
```



See also:

Step 3: [Setting up Client Certificate for Navicat](#)


## Setting up Client Certificate for Navicat

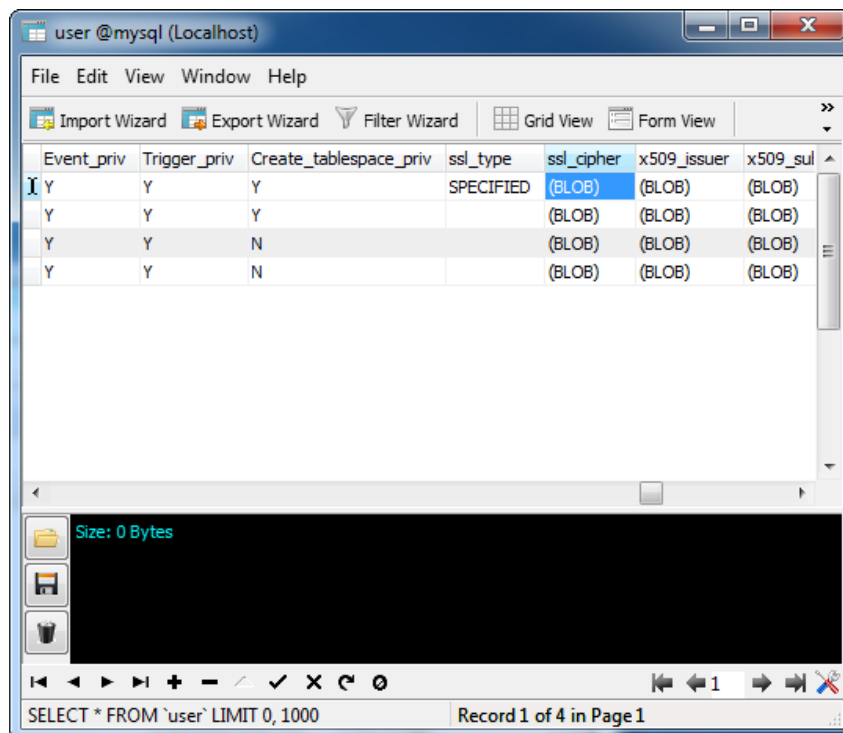
The following instruction guides you through the process of configuring a connection between Navicat and MySQL/PostgreSQL Server using SSL. To successfully establish a SSL connection, please complete [Step 1: Installation of OpenSSL and MySQL/PostgreSQL](#) and [Step 2: Setting up SSL Certificate for MySQL/PostgreSQL](#).

### MySQL



1. Click  or choose File ->  **New Connection** to set up the Connection Properties.
2. Select the SSL tab and enable **Use SSL**.
3. To provide authentication details, fill in the required information:

**Client Key**, **Client Certificate** and **CA Certificate** are usually stored in your Server - `/usr/local/openssl`. Please copy them from the remote server to local computer. **Specified Cipher** (optional) is only required while **ssl\_type** field has been set to "**SPECIFIED**" - [ssl\_type can be found in a system database called "mysql" -> table called "user"]. Example of Specified Cipher is "EDH-RSA-DES-CBC3-SHA" which can be filled in either through the Connection Properties shown above or the "mysql" database -> "user" table -> "ssl\_cipher" blob field shown below.

**Note:** You are allowed to store your Specified Cipher into a text file in order to load  into the "ssl\_cipher" blob field.



## PostgreSQL

1. Click  or choose File ->  **New Connection** to set up the Connection Properties.
2. Select the SSL tab and enable **Use SSL**.
3. Select the **SSL Mode**.
  - require** - only try an SSL connection.
  - verify-ca** - only try an SSL connection, and verify that the server certificate is issued by a trusted CA.
  - verify-full** - only try an SSL connection, verify that the server certificate is issued by a trusted CA and that the server hostname matches that in the certificate.
4. To provide authentication details, enable **Use Authentication** and fill in the required information:

**Client Key**, **Client Certificate** and **CA Certificate** are usually stored in your Server - `/usr/local/openssl`. Please copy them from the remote server to local computer.

**Certificate Revocation List** specifies the file path of the SSL certificate revocation list (CRL).

For PostgreSQL server, OpenSSL supports a wide range of ciphers and authentication algorithms, of varying strength. While a list of ciphers can be specified in the OpenSSL configuration file, you can specify ciphers specifically for use by the database server by modifying `ssl_ciphers` in `postgresql.conf`.

See also:

[Advanced Settings](#)

## Advanced Settings

Customize connection options according to your needs. The detailed description is given below:

### Settings Save Path

When a new connection being established, Navicat will create a subfolder under the Settings Save Path. Most files are stored within this subfolder:

Navicat Objects	Server Type	File Extensions
Query	All	.sql
Export Query Result Profile	MySQL	.npeq
	Oracle	.nopeq
	PostgreSQL	.nppeq
	SQLite	.nlpeq
Export View Result Profile	MySQL	.npev
	Oracle	.nopev
	PostgreSQL	.nppev
	SQLite	.nlpev
Backup	MySQL, PostgreSQL and SQLite	compressed (.psc), uncompressed (.psb)
Backup Profile	MySQL	.npb
	PostgreSQL	.nppb
	SQLite	.nlpb
Report	All	.rtm
Import Wizard Profile	MySQL	.npi
	Oracle	.nopi
	PostgreSQL	.nppi
	SQLite	.nlpi
Export Wizard Profile	MySQL	.npe
	Oracle	.nope
	PostgreSQL	.nppe
	SQLite	.nlpe
Export Materialized View Profile	Oracle	.nopem

Other files are located in the **profiles** directory. To look for the path, choose Tools -> Options -> Miscellaneous -> Profiles Save Path.

Other Profiles	Server Type	File Extensions
Data Transfer	MySQL	.npt
	Oracle	.nopt
	PostgreSQL	.nppt
	SQLite	.nlpt
	Premium (Cross Server)	.napt
Data Synchronization	MySQL	.npd
	Oracle	.nopd
	PostgreSQL	.nppd
	SQLite	.nlpd
Structure Synchronization	MySQL	.nps
	Oracle	.nops
	PostgreSQL	.npps
	SQLite	.nlps
Batch Job	Premium (Cross Server)	.napj
Virtual Grouping	All	vgroup.xml - stores how the objects are categorized.

**Hint:** All your connection settings are stored in registry.

## MySQL

### Encoding

Choose a codepage to communicate with MySQL Server while MySQL character set not being employed.

#### **Keepalive Interval (sec)**

This option allows you to keep the connection with the server alive by pinging it. You can set the period between pings in the edit field.

#### **Use MySQL character set**

This option should be enabled if employing MySQL 4.1 or above.

#### **Use Compression**

This option allows you to use compression protocol. It is used if both client and server support zlib compression, and the client requests compression.

#### **Auto Connect**

With this option on, Navicat automatically open connection with the registered database at application startup.

#### **Use Named Pipe, Socket**

With this option on, Navicat uses socket file for localhost connection.

## Oracle

### Role

Indicate that the database user is connecting with either the **Default**, **SYSOPER** or **SYSDBA** system privilege.

#### **OS Authentication**

With this option on, Oracle Database uses Windows user login credentials to authenticate database users.

#### **Auto Connect**

With this option on, Navicat automatically opens connection with the registered database at application startup.

## PostgreSQL

### **Keepalive Interval (sec)**

This option allows you to keep the connection with the server alive by pinging it. You can set the period between pings in the edit field

### **Auto Connect**

With this option on, Navicat automatically opens connection with the registered database at application startup.

## SQLite

### **Auto Connect**

With this option on, Navicat automatically opens connection with the registered database at application startup.

### **Encrypted**

Enable this option and provide **Password** when connecting to an encrypted SQLite database.

### **Attached Database**

To attach or detach databases in the connection.

## Setting Advanced Database Properties (Available only for MySQL and PostgreSQL Servers)

Set the advanced database properties, which are not obligatory. To start working with advanced database settings, check the **Use Advanced Connections**. The detailed description is given below:

### Show Selected Databases

To show the selected databases in the **close** state in the navigation pane

- Click the preferable databases in the Databases list box, the check box will show as

To show the selected databases in the **open** state in the navigation pane

- Double-click the preferable databases in the Databases list box, the check box will show as

### Add Hidden Database

To add a hidden database

- Click **Add DB to List** button.
- Enter the database name.
- Select the newly added database in the Databases list box.
- Enter **User Name** and **Password** which provide by your ISP.

### Remove Database

To remove a database

- Select the database to remove in the Databases list box.
- Click **Remove DB from List** button.

**Note:** The database will be just removed from the Databases list box, it will still exist in the Server.

See also:

[Advanced Settings](#)

Related topic:

[Connection Settings](#)