

# **Advantages of Server-side Database Auditing**

By SoftTree Technologies, Inc.

## Table of Contents

|  |    |
|--|----|
| Advantages of server-side auditing.....  | 3  |
| Does server-side auditing create a performance hit on the audited databases? .....                     | 3  |
| Why would you want to use server-side auditing instead of so-called non-intrusive auditing tools?..... | 4  |
| What vendors of network-based auditing tools don't want to tell you.....                               | 4  |
| What vendors of transaction log based auditing tools don't want to tell you .....                      | 5  |
| Sybase ASE.....  | 6  |
| Microsoft SQL Server 2000.....   | 7  |
| Microsoft SQL Server 2005.....   | 8  |
| Oracle.....  | 10 |
| DB2.....   | 11 |
| Dictionary .....   | 15 |

## Advantages of server-side auditing

1. Server-side auditing is the only method allowing auditing of every type of database access by any type of user, regardless of whether users are network based or local to the server. Server-side auditing can also audit all types of local access to the database server performed via remote system access protocols such as Telnet, SSH, Windows RDP, Terminal servers, Citrix, Symantec pcAnywhere, just to name a few.
2. Server-side auditing also allows auditing user activities performed using encrypted network protocols (SSH, SSL, etc..) and encrypted database client-server communication protocols (Oracle TCPS, DB2 APPC, ASE SSL encryption, MySQL compression, SQL Server connection encryption, etc..). All these encrypted database communication protocols can be audited just as well as regular non-encrypted protocols.
3. Server-side auditing provides reach audit-time filtering not available in other auditing products
4. Server-side auditing records data to audit trail tables or optionally to external files in a real-time and the recorded data is immediately available for use in reporting and alerting processes. This data can be used for real-time or historical forensic analysis.
5. Audit trails contain data in structured format, with audit information presented as a set of time-stamped events with a rich set of attributes. This information can be easily analyzed using any mainstream database reporting tools and you are not locked to proprietary reporting technologies and canned reports provided with the audit tools, which are usually limited in capabilities and scope.
6. The collected audit events can be easily aggregated and correlated with application and system event logs. This is especially important in web applications that utilize connection pools using a single database user account (kind of a super-user) for every application user. The correlation of database events can be performed in a real time or after the fact, linking application user names to database sessions and changes and unmasking these otherwise anonymous users.
7. The collected server-side audit events can be easily replicated to an enterprise audit repository server where they cannot be altered or modified by personnel who have local access to the audited database servers.

## Does server-side auditing create a performance hit?

Sure, every additional processing on the database server such as auditing, inevitable creates additional I/O and requires additional CPU resources. But if the auditing is setup properly and focused on what is really needed, in other words, if you don't attempt to capture every bit of data and every command executed by every user on a very busy

production server, the performance hit should be really negligible and more than acceptable in regard to the benefits offered by the server-side auditing.

## **Why would you want to use server-side auditing instead of so-called non-intrusive auditing tools?**

Besides the advantages described in the first topic you should consider the following facts:

1. Server-side auditing is the only way to audit complete database server activities and every protocol and type of database access by any type of user, regardless of whether users are network based or local to the server. Server-side auditing is the only solution for monitoring every type of back-door access to the server and the data.
2. No changes or functionality sacrifices are required in any existing applications and systems when server-side auditing is deployed. Use whatever application or database access methods you use or want to use without any changes or restrictions. Run your regular database management tools, data load tools, such as SQL\*Loader, DTS, LOAD TABLE and other directly on the server as you use them now

**Read the following paragraphs to find out how many different methods are available for every major database system to access the data in the database, bypassing the entire network stack and along with it, bypassing all network traffic sniffing tools as well as find out how to avoid leaving any tracks in database transaction logs.**

## **What vendors of network-based auditing tools don't want to tell you**

1. There are many database communication protocols available. Look around; TCP is only one of many. Can your vendors handle non-TCP protocols?
2. Database communications can be and should be encrypted. When communications are encrypted, network based tools and hackers cannot peek into the data traveling on the network between your applications and database servers.
3. Local database access using IPC or shared memory methods cannot be audited from outside of the local system. Virtually every DBA as well as many privileged users use such database access methods!

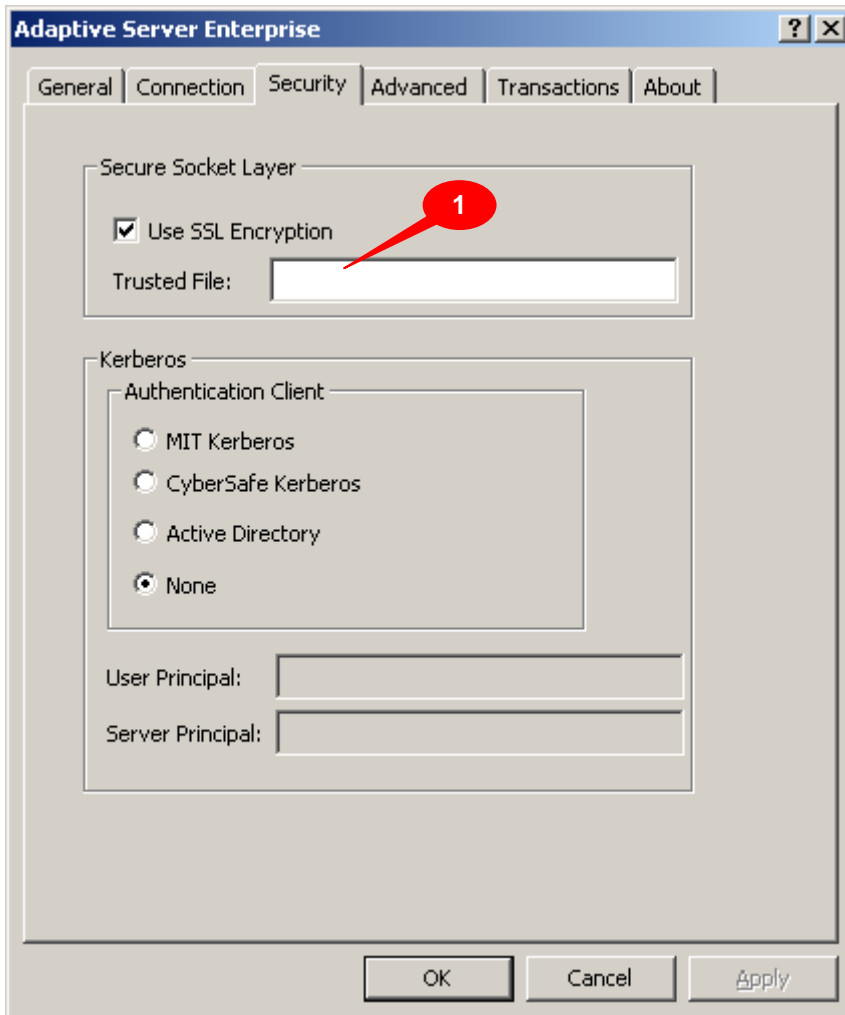
4. Internal database activities and data relations cannot be sniffed or audited from outside. For example, table A can be accessed via view B and that view is referenced in views C, D and F, Each view has some joins and data filters (WHERE or HAVING clauses). If users never reference table A directly in their SQL commands, how can you know when table A is really accessed and which part of it? Network-based side SQL sniffing is no help here.
5. SQL sniffing off the network and parsing the captured traffic data without physical database connection and inside the database user session context is simply unreliable by design. Just think of user impersonation, use of database views and object aliases and other database things that exist specifically to prevent direct user access to the underlying data.
6. Network-based auditing tools cannot tell you which privileges are actually used by users to perform database operations.
7. There is no way to correlate or see the real impact and data-access resulting for each SQL command. Consider, for example, a hacker sending a command to the database that schedules an automated job to save employee data to a file during late night hours. The data is not updated or retrieved over the wire by that command. Moreover, the leak of this sensitive data doesn't even happen in the same user session or same time. The hacker can later pick the created file from the operation system without accessing the database again.

## **What vendors of transaction log based auditing tools don't want to tell you**

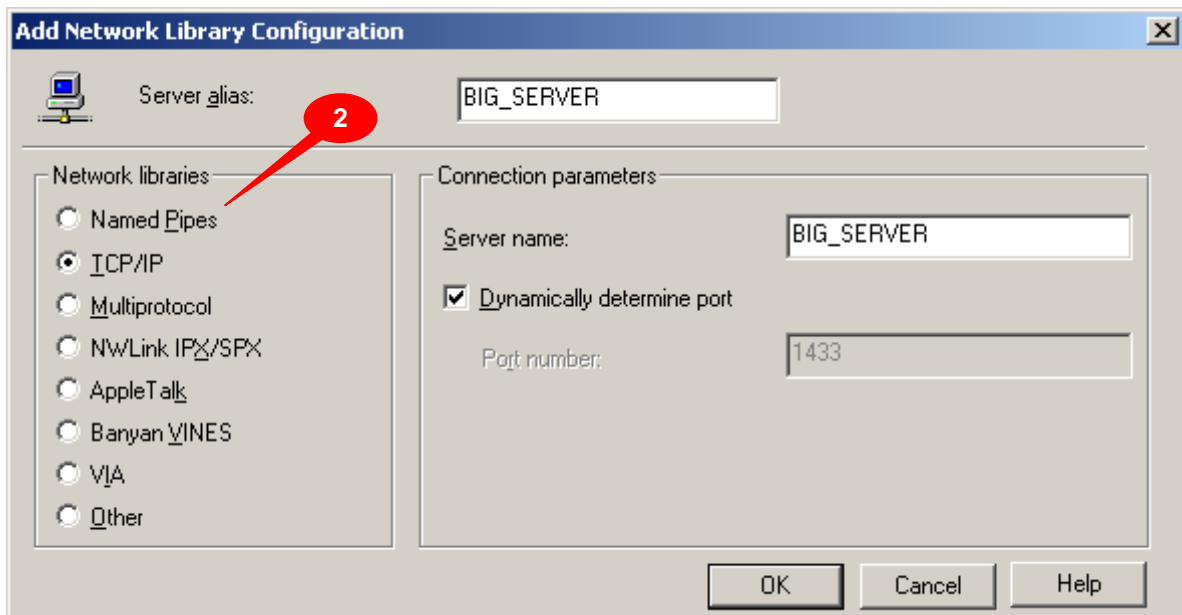
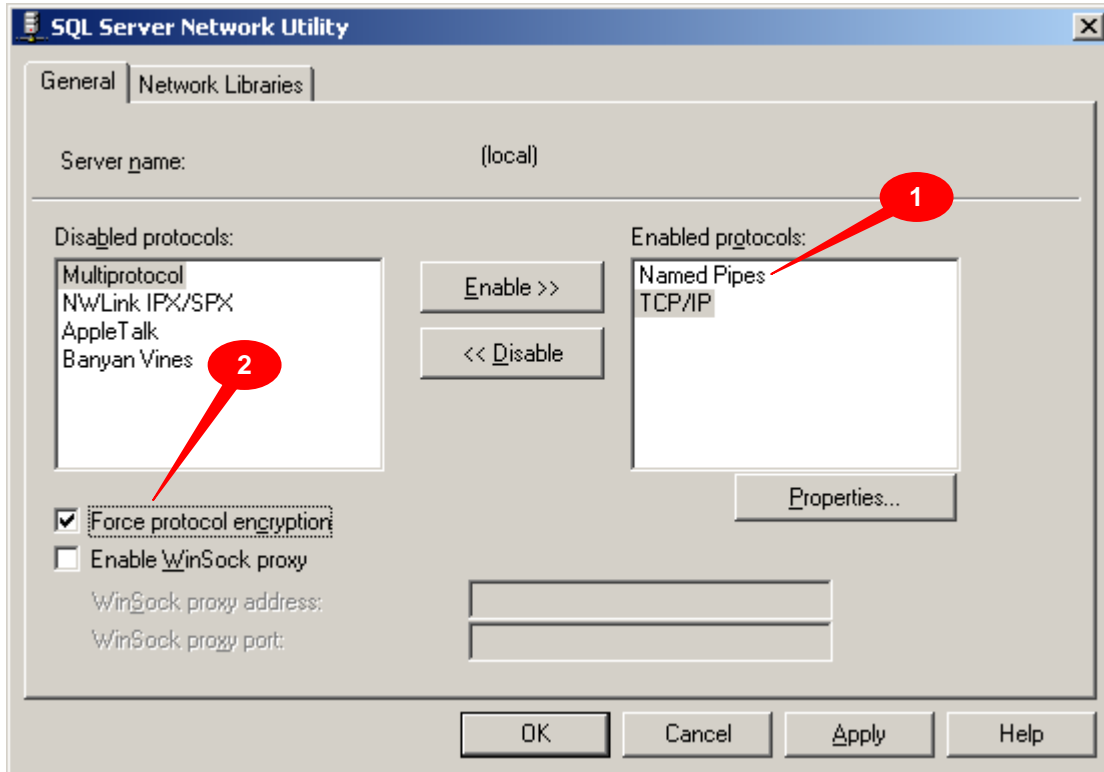
1. Only small subset of user activities is tracked in transaction logs. It is fair to say that less than 10 % of database operations are typically recorded in these logs. Transaction log records only cover database updates, but what about SELECT, EXECUTE and other common commands used to access data in the database?
2. Transaction logs typically do not contain any information about unauthorized activities, making it impossible to backtrack any unauthorized access and update attempts, security violations and other related events considered as a must-have for any system which is subject to auditing for SOX compliance and other government regulations.
3. In case of failed data updates, the database system rolls back all intermediate changes and as a result the associated log data completely vanishes from the transaction logs.
4. Transaction logs are limited in size. Many database systems use log rotation or truncation during backups in which case the logged transaction data simply disappears from the transaction logs. As a result, transaction logs become unusable for backtracking old activities and for performing accurate analysis of user activity patterns.

## Appendix A: Use of non TCP-based protocols and data encryptions methods

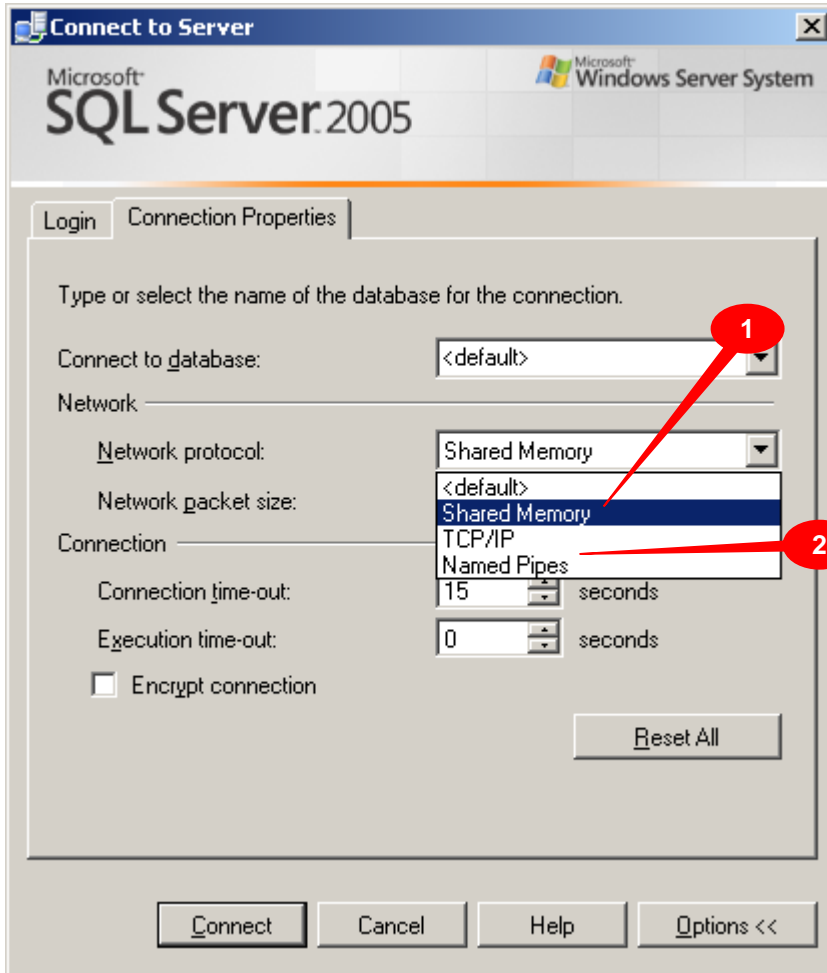
### Sybase ASE



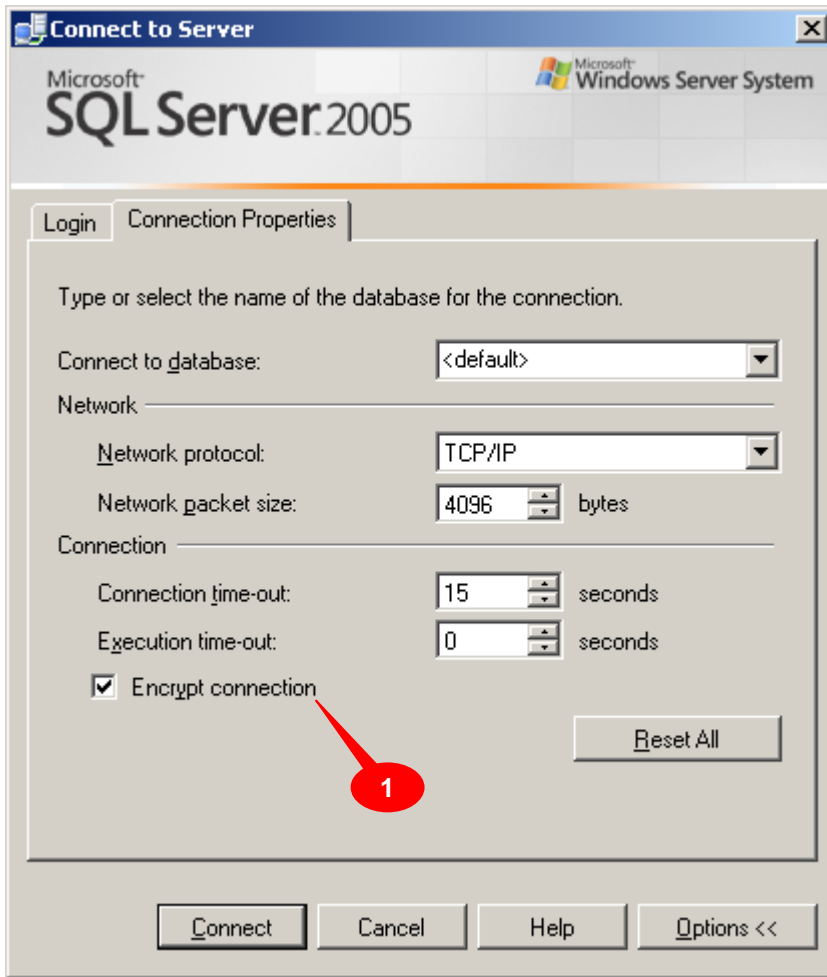
## Microsoft SQL Server 2000



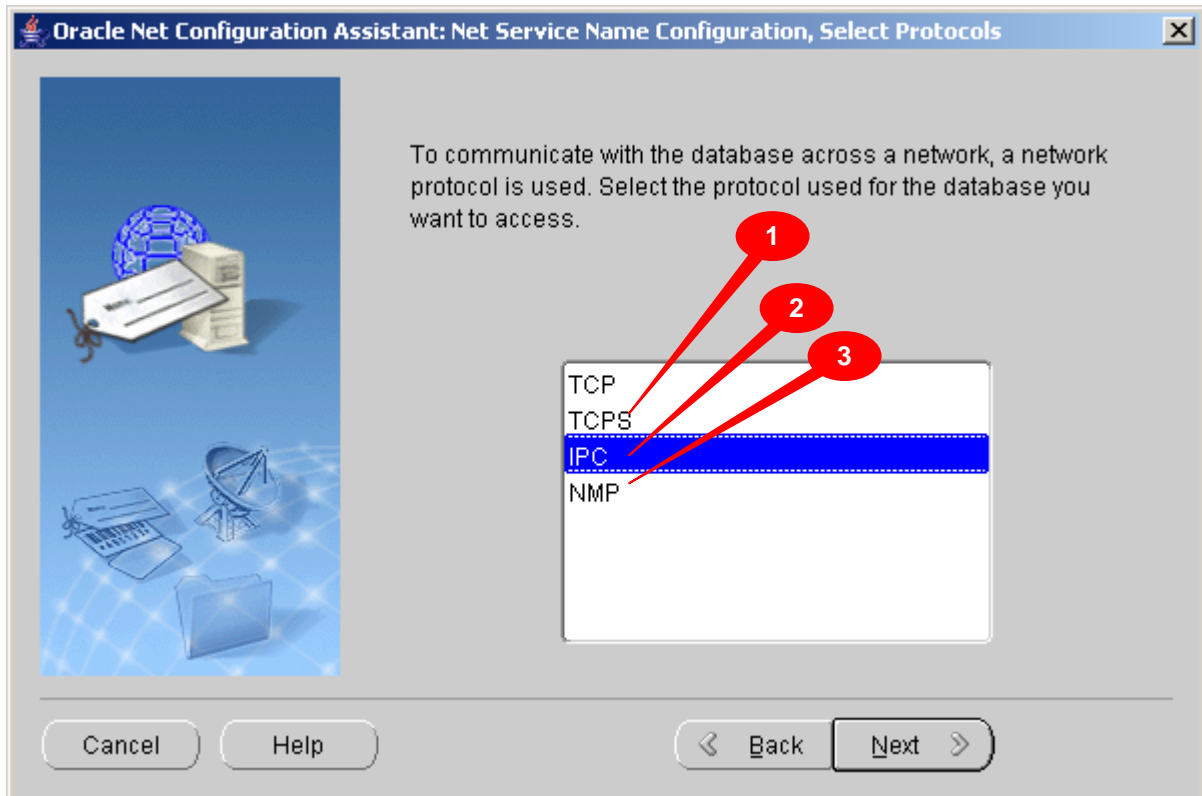
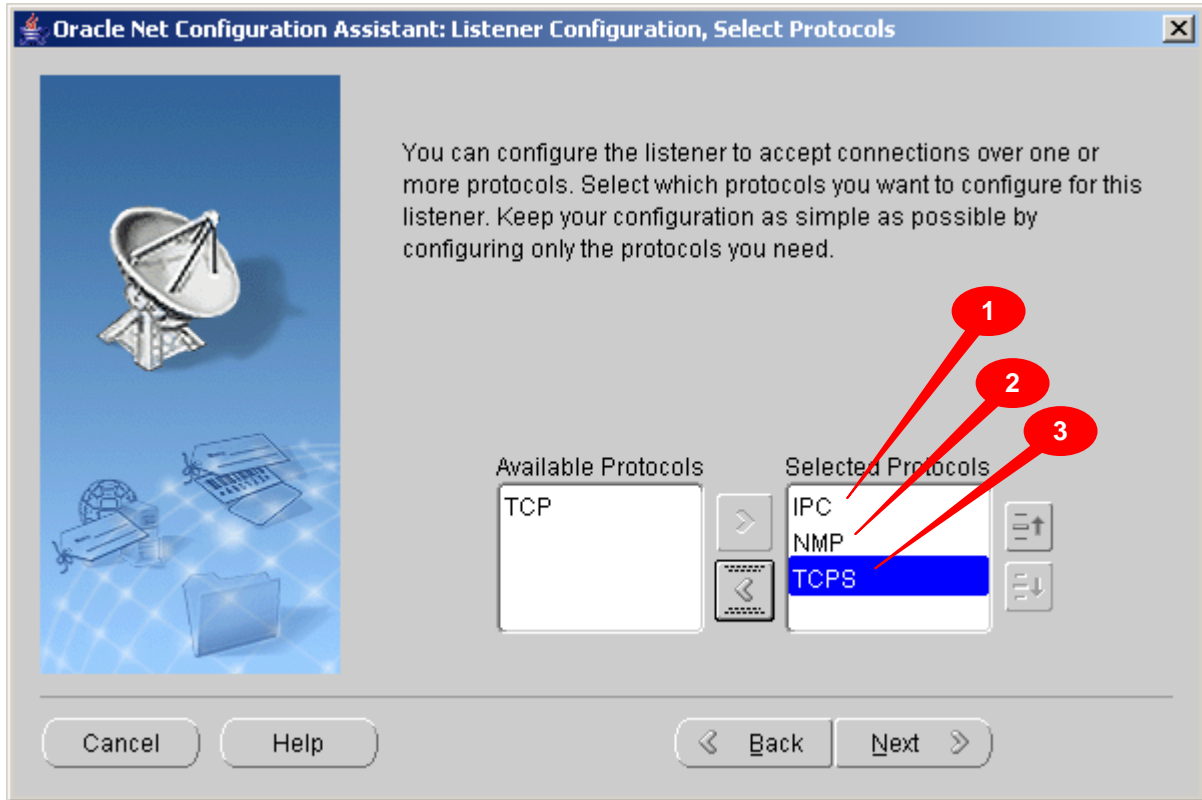
## Microsoft SQL Server 2005



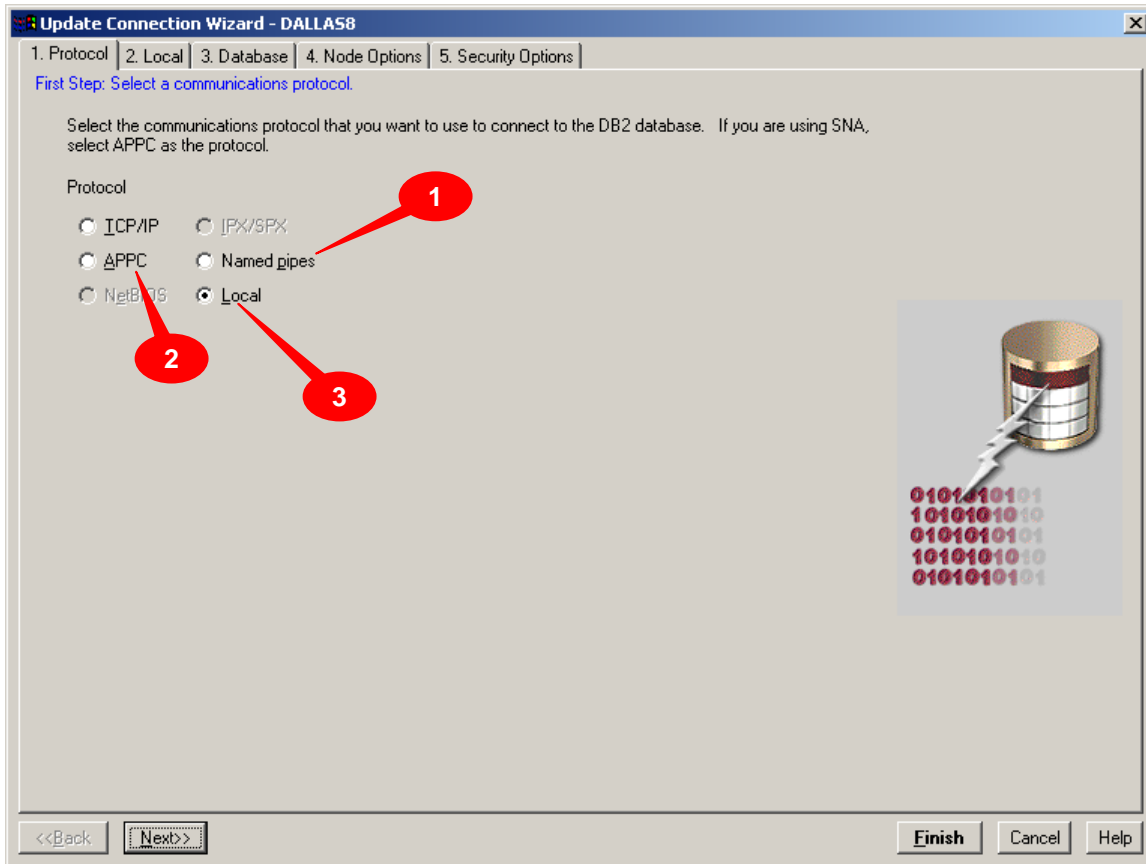




## Oracle



## DB2



**Update Connection Wizard - DALLAS8**

1. Protocol | 2. TCP/IP | 3. Database | 4. Node Options | 5. Security Options

Last Step: (Optional) Specify the security options.

Specify the security options to determine the method of validation. Client authentication occurs on the client. Server authentication occurs on the server or gateway. Host or AS/400 authentication occurs on the host or AS/400 system. DCE authentication occurs on a DCE server, which must be specified in the Server principal name field. Kerberos authentication occurs on a Kerberos server, which must be specified in the Target principal name field.

Use authentication value specified in the server's DBM configuration

Client authentication (CLIENT)

Server authentication (SERVER)

Enable encryption

Host or AS/400 authentication (DCS)

Enable encryption

DCE authentication (DCE)


Server principal name:

Kerberos authentication (KERBEROS)

Target principal name:

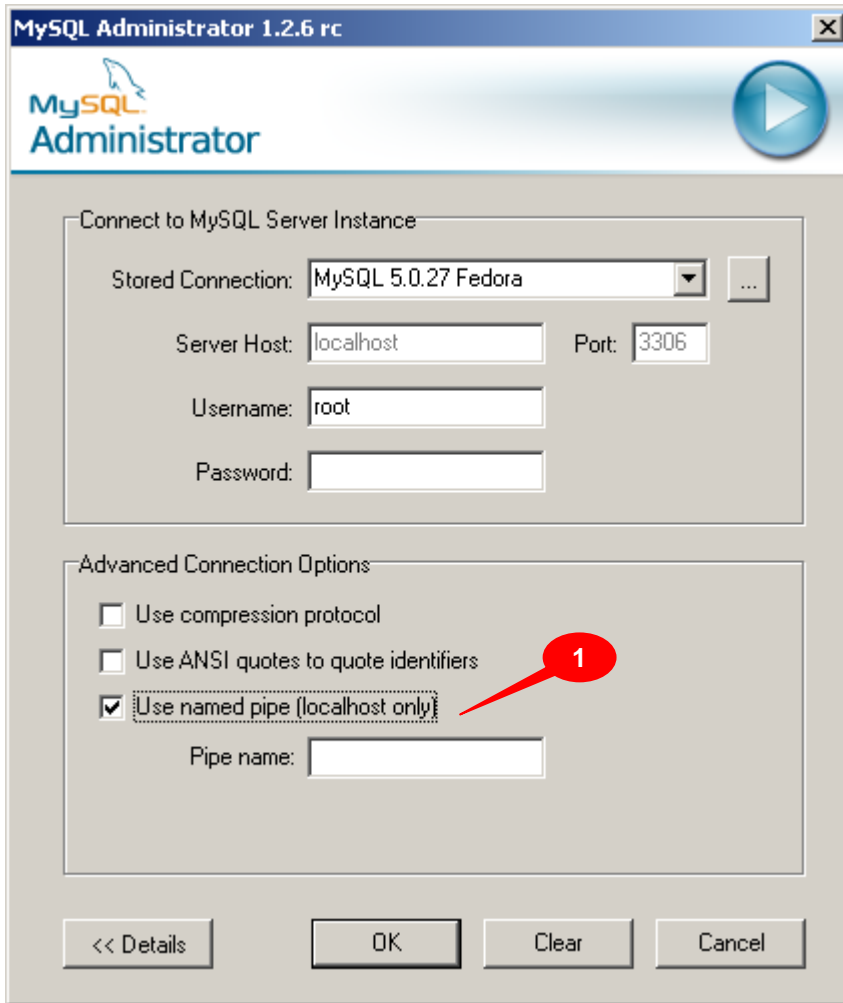
TCP/IP security

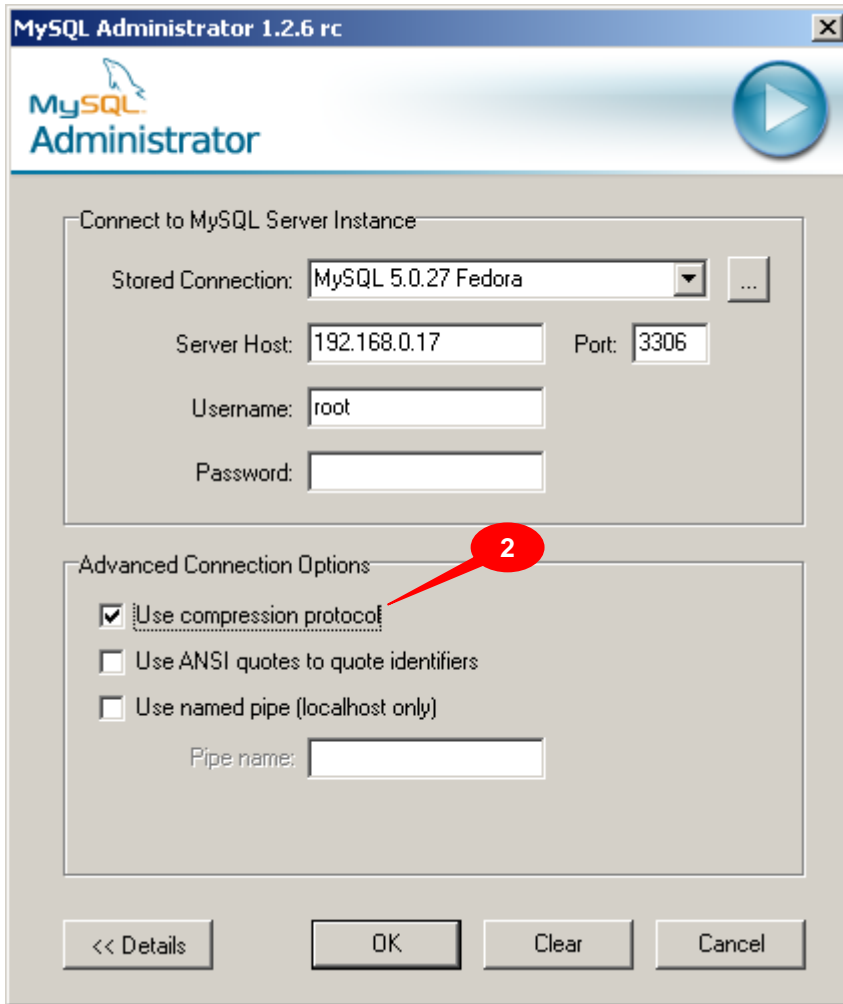
Enable TCP/IP SOCKS security



<<Back | Next>> | Finish | Cancel | Help

# MySQL





## Appendix B: Dictionary

A number of common abbreviations and technical terms were used on the screenshots provided in this document. Here is brief description of these terms.

**IPC** - the Inter-Process Communication (IPC) protocol is used for client applications that are on the same computer as the Oracle listener. IPC doesn't use network stack and allows client communicate with the local server directly.

**Shared Memory** this is just another kind of IPC.

**Named Pipes**– the Named Pipes is used for client applications that are not using TCP/IP. Named Pipes method is available on both Windows and Unix systems.

**NMP** - this is essentially the same as Named Pipes

**TCPS** - the TCP/IP with Secure Sockets Layer (SSL) protocol enables a client application on a client to communicate with remote databases through TCP/IP and SSL using SSL encryption applied to the messages they exchange.

**TCP/IP SSL** – this is the same as TCPS