


Keeping Your Data Secure

About this chapter

This chapter describes Adaptive Server Anywhere features that help to make your database secure.

Many of these features are described in more detail elsewhere in the documentation, and for such features, pointers to the relevant places are provided.

Database administrators are responsible for data security. In this chapter, unless otherwise noted, you require DBA authority to carry out the tasks described.

 User IDs and permissions are major security-related topics. For information on these topics, see "Managing User IDs and Permissions" on page 575.






Contents

Topic	Page
Security features overview	602
Security tips	603
Controlling database access	605
Controlling the tasks users can perform	607
Auditing database activity	608
Running the database server in a secure fashion	612

Security features overview

Databases may contain information that is proprietary, confidential, or private. For this reason, it can be important to ensure that the database and the data in it are designed for security.

Adaptive Server Anywhere has several features to assist in building a secure environment for your data:

- ◆ **User identification and authentication** These control who can gain access to a database.
 For information on these subjects, see "Creating new users" on page 580.
- ◆ **Discretionary access control features** These features control the actions a user is able to carry out when connected to a database.
 For more information, see "Database permissions overview" on page 576.
- ◆ **Auditing** By turning on auditing, you can maintain a record of actions on the database.
 For more information, see "Auditing database activity" on page 608.
- ◆ **Database server options** When you start the database server, you control who can carry out operations such as loading databases.
 For more information, see "Controlling permissions from the command line" on page 11.
- ◆ **Views and stored procedures** Views and stored procedures allow you to tune the data that a user can access and the operations a user can execute.
 For more information, see "Using views and procedures for extra security" on page 593.

This chapter describes auditing, and collects together overviews of the other security features, providing pointers to where they are discussed in more detail.

Security tips

There are many actions you can take as database administrator can take to improve the security of your data.

- ◆ **Change the default user ID and password** The default user ID and password for a newly created database is **DBA** and **SQL**. You should change this password before deploying the database.
- ◆ **Require long passwords** You can set the `MIN_PASSWORD_LENGTH` public option to disallow short (and therefore easily guessed) passwords.

☞ For information, see "MIN_PASSWORD_LENGTH option" on page 163 of the book *Adaptive Server Anywhere Reference Manual*.

- ◆ **Restrict DBA authority** DBA authority is very powerful. Users with DBA authority can see and do anything in the database. You should grant DBA authority only to users who absolutely require it.

Consider giving users with DBA authority two user IDs, one with DBA authority and one without, so that they can connect as DBA only when necessary.

- ◆ **Drop external system functions** The following external functions present possible security risks: `xp_cmdshell`, `xp_startmail`, `xp_sendmail`, and `xp_stopmail`.

The `xp_cmdshell` procedure allows users to cause the server to execute operating system commands or programs.

The e-mail commands allow users to have the server send e-mail composed by the user. Malicious users could use either the e-mail or command shell procedures to perform operating-system tasks with authorities other than they have been given by the operating system. In a security-conscious environment, these functions should be dropped.

☞ For information on dropping procedures, see "DROP statement" on page 451 of the book *Adaptive Server Anywhere Reference Manual*.

- ◆ **Protect your database files** The database file, log files, dbspace files, and write files should be protected from unauthorized access. They should not be stored within a shared directory or volume.
- ◆ **Protect your database software** The Adaptive Server Anywhere software should be similarly protected. Users should be given access only to the applications, DLLs, and other resources that they require.

- ◆ **Run the database server as a service or a daemon** On Windows NT, the database server should be run as an NT service so that unauthorized users cannot shut it down or gain access to the database or log files. On UNIX, running the server as a daemon serves a similar purpose.

☞ For more information, see "Running the server outside the current session" on page 18.

Controlling database access

By assigning user IDs and passwords, the database administrator controls who can gain access to a database. By granting permissions to each user ID, the database administrator controls what tasks each user can carry out when connected. This section describes the features available for controlling database access.

Permission scheme is based on user IDs

When users log into the database, they have access to all database objects that meet *any* of the following criteria:

- ◆ The object was created by that user.
- ◆ The user was explicitly granted permission on the object.
- ◆ A group to which the user belongs was explicitly granted permission on the object.

The user cannot access any database object that does not meet these criteria. In short, users can access only objects that they own or to which access has been explicitly granted.

For more information, see the following:

- ◆ "Managing User IDs and Permissions" on page 575.
- ◆ "CONNECT statement" on page 381 of the book *Adaptive Server Anywhere Reference Manual*.
- ◆ "GRANT statement" on page 484 of the book *Adaptive Server Anywhere Reference Manual*.
- ◆ "REVOKE statement" on page 536 of the book *Adaptive Server Anywhere Reference Manual*.



Using integrated logins

Integrated logins allow users to use a single login name and password to log into the Windows NT operating system and into a database. An external login name is associated with a database user ID. When a user attempts an integrated login, the operating system tells the server who the user is, and the server logs the user in as the associated database user ID. No login name or password are required, since the user provided both in order to log into the operating system. There are some security implications of integrated logins to consider

For more information see the following

- ◆ "Using integrated logins" on page 58.
- ◆ "Security concerns: unrestricted database access" on page 62.
- ◆ "LOGIN_MODE option" on page 161 of the book *Adaptive Server Anywhere Reference Manual*.

Increasing password security

	<p>Passwords are an important part of any database security system. To be secure, passwords must not be easy to guess, and they must not be easily accessible on users' hard drives or other locations.</p>
Restricting password length	<p>By default, passwords can be any length. For greater security, you can enforce a minimum length requirement on all new passwords. You do this by setting the MIN_PASSWORD_LENGTH database option to a value greater than zero. The following statement enforces passwords to be at least 8 bytes long.</p> <pre>SET OPTION PUBLIC.MIN_PASSWORD_LENGTH = 8</pre> <p> For more information, see "MIN_PASSWORD_LENGTH option" on page 163 of the book <i>Adaptive Server Anywhere Reference Manual</i>.</p>
Encrypt the passwords	<p>As passwords are the key to accessing databases, it is important that they not be easily available to unauthorized people in a security-conscious environment.</p> <p>When you create an ODBC data source, or a Sybase Central connection profile, you can optionally include a password. Avoid including passwords for greater security. If you do include a password in the data source, check the box to encrypt the password.</p> <p> For information on creating ODBC data sources, see "Creating an ODBC data source" on page 42.</p>

Controlling the tasks users can perform

Users can access only those objects to which they have been granted access.

Granting permission on an object to another user is done using the GRANT statement. It is also possible to grant a user a grant option to an object, which allows that user to pass on the permissions to other users.

The GRANT statement is also used to give more general permissions to users. Granting CONNECT permissions to a user is used to create users and to change their passwords. Granting RESOURCE to a user is required for the user to create tables, views, procedures, etc. Granting DBA to a user gives that user the ability to see and do anything in the database. The DBA would also use the GRANT statement to create and administer groups.

The REVOKE statement is the opposite of the GRANT statement—any permission that GRANT has explicitly given, REVOKE can take away. Revoking CONNECT from a user will remove the user from the database, including all objects owned by that user.

Negative permissions

Adaptive Server Anywhere does not support **negative permissions**. This means that you cannot revoke a permission that was not explicitly granted.

For example, suppose user **bob** is a member of a group called **sales**. If a user grants DELETE permission on a table T to sales, then bob can delete rows from T. If you want to prevent bob from deleting from T, you cannot simply execute a REVOKE DELETE on T from bob, since the DELETE ON T permission was never granted directly to bob. In this case, you would have to revoke bob's membership in the sales group.

For more information see:

- ◆ "GRANT statement" on page 484 of the book *Adaptive Server Anywhere Reference Manual*.
- ◆ "REVOKE statement" on page 536 of the book *Adaptive Server Anywhere Reference Manual*.

Designing database objects for security

Views and stored procedures provide alternative ways of tuning the data users can access and the tasks they can perform.

For more information on these features, see:

- ◆ "Benefits of procedures and triggers" on page 223.
- ◆ "Using views and procedures for extra security" on page 593.


Auditing database activity

Auditing is a way of keeping track of the activity performed on a database. The record of activities is kept in the transaction log. By turning on auditing, the DBA increases the amount of data saved in the transaction log to include the following:

- ◆ All login attempts (successful and failed), including the terminal ID.
- ◆ Accurate timestamps of all events (to a resolution of milliseconds)
- ◆ All permissions checks (successful and failed), including the object on which the permission was checked (if applicable)
- ◆ All actions that require DBA authority.

The transaction log

Each database has an associated transaction log file. The transaction log is used for database recovery. It is a record of transactions executed against a database.

 For information about the transaction log, see "The transaction log" on page 557.

The transaction log stores all executed data definition statements, and the user ID that executed them. It also stores all updates, deletes, and inserts and which user executed those statements. However, this is insufficient for some auditing purposes. By default, the transaction log does not contain the time of the event, just the order in which events occurred. It also contains no failed events, nor select statements.

Turning on auditing

The database administrator can turn on **auditing**. Auditing adds security-related information to the transaction log.

Auditing is disabled by default. To enable auditing on a database, the DBA must set the value of the public option AUDITING to ON. DBA permissions are required to set this option. Auditing then remains enabled until explicitly disabled, by setting the value of the AUDITING option to OFF.

❖ To turn on auditing:

- 1 Ensure that your database is upgraded to at least version 6.0.2.
- 2 If you had to upgrade your database, create a new transaction log.
- 3 Execute the following statement:

```
SET OPTION PUBLIC.AUDITING = 'ON'
```


For more information, see "AUDITING option" on page 140 of the book *Adaptive Server Anywhere Reference Manual*.

Retrieving audit information

You can use the Log Translation utility to retrieve audit information. You can access this utility from Sybase Central or from the command line. It operates on a transaction log to produce a SQL script containing all of the transactions, along with some information on what user executed each command. By using the `-g` option, *dbtran* includes more comments containing the auditing information.

To ensure a complete and readable audit record, the `-g` option automatically sets the following switches:

- ◆ **-d** Display output in chronological order.
- ◆ **-t** Include trigger-generated operations in the output.
- ◆ **-a** Include rolled back transactions in the output.

The Log Translation Utility can be run against a running database server or against a database log file

❖ To retrieve auditing information from a running database server:

- 1 With the database server running, execute the following statement at a system command prompt.

```
dbtran -g -c "uid=dba;pwd=sql;..." -n asademo.sql
```

The user ID must have DBA authority.

For information about connection strings, see "Connection parameters" on page 46.

❖ To retrieve auditing information from a transaction log file:

- 1 Ensure the log file is not in use by closing down the database server.
- 2 At a system command prompt, execute the following statement to place the information from the file *asademo.log* and places it in the file *asademo.sql*.

```
dbtran -g asademo.log
```

The `-g` command-line option includes auditing information in the output file.

For more information see "The Log Translation utility" on page 98 of the book *Adaptive Server Anywhere Reference Manual*.

Adding audit comments

You can add comments to the audit trail using the **sa_audit_string** system stored procedure. You must have DBA permissions to call this procedure. It takes a single argument, which is a string of up to 200 bytes.

For example:

```
call sa_audit_string( 'Started audit testing here.' )
```

This comment is stored in the transaction log as an audit statement.

An auditing example

This example shows how the auditing feature records attempts to access unauthorized information.

- 1 As database administrator, turn on auditing.

You can do this from Sybase Central as follows:

- ◆ Connect to the ASA 6.0 Sample data source. This connects you as the **DBA** user.
- ◆ Right-click on the **asademo** database icon, and select Set Options from the popup menu.
- ◆ Select Auditing from the list of options, and enter the value ON in the Public Setting box. Click Set Permanent Now to set the option and Close to exit.

Alternatively, you can use Interactive SQL. Connect to the sample database from Interactive SQL, as user ID **DBA** using password **SQL** and execute the following statement:

```
SET OPTION PUBLIC.AUDITING = 'ON'
```

- 2 Add a user to the sample database, named **BadUser**, with password **BadUser**. You can do this from Sybase Central. Alternatively, you can use Interactive SQL and enter the following statement:

```
GRANT CONNECT TO BadUser  
IDENTIFIED BY 'BadUser'
```

- 3 Using Interactive SQL, connect to the sample database as **BadUser** and attempt to access confidential information in the **employee** table with the following query:

```
SELECT emp_lname, salary  
FROM dba.employee
```

You receive an error message: do not have permission to select from employee.

- 4 From a command prompt, change directory to your Adaptive Server Anywhere installation directory, which holds the sample database, and execute the following command:

```
dbtran -g -c "dsn=ASA 6.0 Sample" -n asademo.sql
```

This command produces a file named *asademo.sql*, containing the transaction log information and a set of comments holding audit information. The lines indicating the unauthorized **BadUser** attempt to access the employee table are included in the file as follows:

```
--AUDIT-1001-0000287812 -- 1999/02/11 13:59:58.765
Checking Select permission on employee - Failed
--AUDIT-1001-0000287847 -- 1999/02/11 13:59:58.765
Checking Select permission on employee(salary) -
Failed
```

- 5 Restore the sample database to its original state so that other examples you try in this documentation give the expected results.

Connect as the DBA user, and carry out the following operations:

- ◆ Revoke Connect privileges from the user ID **BadUser**.
- ◆ Set the PUBLIC.AUDITING option to 'OFF'.

Auditing actions outside the database server

Some database utilities act on the database file directly. In a secure environment, the database files should not be accessible to any other than trusted users.

In order to provide auditing of actions directly on the file, under Windows NT only, any use of *dbtran*, *dbwrite*, and *dblog* generates Windows NT Application Events, which can be viewed in the Windows NT Event Viewer.

The user's name and program name are stored in the Event log.

The Application Events are generated only if the database has the AUDITING option set to ON.

Running the database server in a secure fashion

There are several security features that you can set when starting the database server or during server operation. These include the following:

- ◆ Controlling who can start and stop databases, who can create and delete database files, and who can stop the server.

☞ For more information on controlling permissions from the database server command line, see "Controlling permissions from the command line" on page 11.

- ◆ Encrypting client/server communications over the network.

For greater security, you can force client/server network communications to be encrypted as they pass over the network.

Encrypting client/server communications

You can set client/server encryption when you start the database server or, from one client, in the client connection parameters.

❖ To force encryption of client/server communications from the server:

- ◆ Start the database server using the `-e` command-line option. For example:

```
dbsrv6 -e -x tcpip asademo.db
```

☞ For a complete listing of database server command-line options, see "The database server" on page 12 of the book *Adaptive Server Anywhere Reference Manual*.

❖ To force encryption of client/server communications from a particular client:

- ◆ Add the Encryption connection parameter to your connection string.

```
...UID=dba;PWD=sql;ENC=YES;...
```

You can also set this parameter can be set on the Network tab of the connection dialog box and the ODBC data source dialog box.

☞ For more information, see "Encryption connection parameter" on page 49 of the book *Adaptive Server Anywhere Reference Manual*.