CHAPTER 5

The Open Client Interface

About this chapter

This chapter describes the Open Client programming interface for Adaptive Server Anywhere.

The primary documentation for Open Client application development is the Open Client documentation, available from Sybase. This chapter describes features specific to Adaptive Server Anywhere, but is not an exhaustive guide to Open Client application programming.

Contents

What you need to build Open Client applications

Open Client applications require the Open Client components that you can optionally install with Adaptive Server Anywhere.

Ensure your database is compatible

You can access any Adaptive Server Anywhere database using the Sybase Open Client programming interface. However, you should ensure that your database is created for maximum compatibility with Sybase Adaptive Server Enterprise.

☞ For more information about making your database as compatible as possible with Adaptive Server Enterprise, see "Configuring databases for Transact-SQL compatibility" on page 792 of the book *Adaptive Server Anywhere User's Guide*.

Maximum compatibility not required for Replication Server
If you are using your database strictly for Replication Server, it is not critical to deviate from the default settings. If you want to use other Open Client applications, such as OmniConnect, then it is important to use the Adaptive Server Enterprise compatibility settings.

Data type mappings

Open Client has its own internal data types, which differ in some details from those available in Adaptive Server Anywhere. For this reason, Adaptive Server Anywhere internally maps some data types between those used by Open Client applications and those available in Adaptive Server Anywhere.

Data type mapping between Open Client and Adaptive Server Anywhere

The following table illustrates the mapping of data types performed by Adaptive Server Anywhere between internal Adaptive Server Anywhere data types and Open Client data types.

The base data type column is for information only. Applications receive the data type information as in the ASA data type column. The base data type is hidden from the Open Client application.

Open Client data type	ASA data type	ASA base data type
binary (x)	binary (x)	
bit	bit	
char	char	
datetime	datetime	timestamp
smalldatetime	smalldatetime	timestamp
decimal	decimal	
numeric	numeric	
real	real	
float	double	
image	long binary	
tinyint	tinyint	
smallint	smallint	
int	int	
bigint	bigint	
money	money	decimal (19,4)
smallmoney	smallmoney	decimal (10,4)

Open Client data type	ASA data type	ASA base data type
text	long varchar	
varchar	varchar	
varbinary	varbinary	
timestamp	timestamp	
All others	Error	

Mapping of Adaptive Server Anywhere data types

The following table lists the mappings of data types supported in Adaptive Server Anywhere that have no direct counterpart in Open Client.

ASA data type	Open Client data type
unsigned short	smallint
unsigned int	int
unsigned bigint	bigint
date	smalldatetime
time	smalldatetime
serialization	longbinary
java	longbinary
string	varchar
timestamp struct	datetime

Range limitations in data type mapping

Some data types have different ranges in Adaptive Server Anywhere than in Open Client. In such cases, overflow errors can occur during retrieval or insertion of data.

The following table lists Open Client application data types that can be mapped to Adaptive Server Anywhere data types, but with some restriction in the range of possible values.

In most cases, the Open Client data type is mapped to a Adaptive Server Anywhere data type that has a greater range of possible values. As a result, it is possible to pass a value to Adaptive Server Anywhere that will be accepted and stored in a database, but one that is too large to be fetched by an Open Client application.

Data type	Implementation	Lower range	Upper range
MONEY	Open Client	-922,337,203 ,685,477.5808	922,337,203 ,685,477.5807
MONEY	Adaptive Server Anywhere	-1e15 + 0.0001	1e15 - 0.0001
SMALLMONEY	Open Client	-214,748.3648	214,748.3647
DECIMAL (10,4)	Adaptive Server Anywhere	-1e6 + 0.0001	1e6 - 0.0001
TIMESTAMP	Open Client	N/A	N/A
DATETIME	Open Client	Jan 1, 1753	Dec 31, 9999
SMALLDATETIME	Open Client	Jan 1, 1900	June 6, 2079
DATETIME	Adaptive Server Anywhere	Jan 1, 0001	Dec 31, 9999
BIT	Open Client	0	1
TINYINT	Adaptive Server Anywhere	0	255

Example

For example, the Open Client MONEY and SMALLMONEY data types do not span the entire numeric range of their underlying Adaptive Server Anywhere implementations. Therefore it is possible to have a value in a Adaptive Server Anywhere column which exceeds the boundaries of the Open Client data type MONEY. When the client fetches any such offending values via Adaptive Server Anywhere, an error is generated (Error 22: Invalid data format. Possible overflow or underflow).

Timestamps

The Adaptive Server Anywhere implementation of the Open Client TIMESTAMP data type, when such a value is passed in Adaptive Server Anywhere is different from that of Adaptive Server Enterprise. In Adaptive Server Anywhere the value is mapped to the Adaptive Server Anywhere DATETIME data type. The default value is NULL in Adaptive Server Anywhere, and no guarantee is made of its uniqueness. By contrast, Adaptive Server Enterprise ensures that the value is monotonically increasing in value, and so is unique.

By contrast, the Adaptive Server Anywhere TIMESTAMP data type contains year, month, day, hour, minute, second and fraction of second information, and the DATETIME data type has a greater range of possible values than the Open Client data types that are mapped to it by Adaptive Server Anywhere.

Other restrictions

Adaptive Server Adaptive Server Anywhere defines SYSNAME as VARCHAR (30) and does not allow the Open Server client application to specify the length of the data field.

Data types with limited compatibility

The following tables lists Open Client data types for which there is nosupport in Adaptive Server Anywhere.

Open Client application data type	Problem
LONG	Not supported
SENSITIVITY	Not supported
BOUNDARY	Not supported
VOID	Not supported

Using SQL in Open Client applications

This section provides a very brief introduction to using SQL in Open Client applications, with a particular focus on Adaptive Server Anywhere-specific issues.

☞ For an introduction to the concepts, see "Using SQL in Applications" on page 199 of the book *Adaptive Server Anywhere User's Guide*. For a complete description, see your Open Client documentation.

Executing SQL statements

You send SQL statements to a database by including them in Client Library function calls. For example, the following pair of calls executes a DELETE statement:

```
ret = ct_command(cmd, CS_LANG_CMD,
                 "DELETE FROM employee
                  WHERE emp_id=105"
                CS_NULLTERM,
                CS_UNUSED);
ret = ct_send(cmd);
```

The **ct_command** function is used for a wide range of purposes.

Using prepared statements

The **ct_dynamic** function is used to manage prepared statements. This function takes a *type* parameter which describes the action you are taking.

❖ **To use a prepared statement in Open Client:**

1. Prepare the statement using the **ct_dynamic** function, with a CS_PREPARE *type* parameter.
2. Set statement parameters using **ct_param**.
3. Execute the statement using **ct_dynamic** with a CS_EXECUTE *type* parameter.
4. Free the resources associated with the statement using **ct_dynamic** with a CS_DEALLOC *type* parameter.

☞ For more information on using prepared statements in Open Client, see your Open Client documentation

Using cursors

The **ct_cursor** function is used to manage cursors. This function takes a *type* parameter which describes the action you are taking.

Supported cursor types

Not all the types of cursor that Adaptive Server Anywhere supports are available through the Open Client interface. You cannot use scroll cursors, dynamic scroll cursors, and insensitive cursors through Open Client.

Uniqueness and updateability are two properties of cursors. Cursors can be unique (each row carries primary key or uniqueness information, regardless of whether it is used by the application) or not. Cursors can be read only or updateable. If a cursor is updateable and not unique, performance may suffer, as no prefetching of rows is done in this case, regardless of the CS_CURSOR_ROWS setting (see below).

The steps in using cursors

In contrast to some other interfaces, such as Embedded SQL, Open Client associates a cursor with a SQL statement expressed as a string. Embedded SQL first prepares a statement, and then the cursor is declared using the statement handle.

❖ **To use cursors in Open Client:**

1. To declare a cursor in Open Client, you use **ct_cursor** with CS_CURSOR_DECLARE as the *type* parameter.
2. After declaring a cursor, you can control how many rows are prefetched to the client side each time a row is fetched from the server by using **ct_cursor** with CS_CURSOR_ROWS as the *type* parameter.

   Storing prefetched rows at the client side cuts down the number of calls to the server, and this improves overall throughput as well as turnaround time. Prefetched rows are not immediately passed on to the application, they are stored in a buffer at the client side ready for use.

   The setting of the PREFETCH database option controls prefetching of rows for other interfaces. It is ignored by Open Client connections. The CS_CURSOR_ROWS setting is ignored for non-unique, updateable cursors.
3. To open a cursor in Open Client, you use **ct_cursor** with CS_CURSOR_OPEN as the *type* parameter.
4. To fetch each row in to the application, you use **ct_fetch**.
5. To close a cursor, you use ct_cursor with CS_CURSOR_CLOSE.

6 In Open Client, you also need to deallocate the resources associated with a cursor. You do this using ct_cursor with CS_CURSOR_DEALLOC. You can also use CS_CURSOR_CLOSE with the additional parameter CS_DEALLOC to carry out these operations in a single step.

In Embedded SQL, cursors are not deallocated. Because Embedded SQL cursors are associated with prepared statements, the resources are associated with the statement itself, and you need to drop the statement in order to free the resources.

Modifying rows through a cursor

With Open Client, you can delete or update rows in a cursor, as long as the cursor is for a single table. The user must have permissions to update the table and the cursor must be marked for update.

❖ **To modify rows through a cursor:**

- Instead of carrying out a fetch, you can delete or update the current row of the cursor using ct_cursor with CS_CURSOR_DELETE or CS_CURSOR_UPDATE, respectively.

You cannot insert rows through a cursor in Open Client applications.

Describing query results in Open Client

Open Client handles result sets in a different way from some other Adaptive Server Anywhere interfaces.

In Embedded SQL and ODBC, you **describe** a query or stored procedure in order to set up the proper number and types of variables to receive the results. The description is done on the statement itself.

In Open Client, you do not need to describe a statement. Instead, each row returned from the server can carry a description of its contents. If you use `ct_command` and `ct_send` to execute statements, you can use the `ct_results` function to handle all aspects of rows returned in queries.

If you do not wish to use this row-by-row method of handling result sets, you can use `ct_dynamic` to prepare a SQL statement, and use `ct_describe` to describe its result set. This corresponds more closely to the describing of SQL statements in other interfaces.

Open Client event handling

Sybase Client Library applications send requests to Adaptive Server Anywhere. Each request triggers an **event**, and Adaptive Server Anywhere executes a routine to handle that event. The routine that handles an event is called an **event handler**.

Adaptive Server Anywhere support is described in terms of these events: the events to which it can respond, and the ways in which it responds to those events.

Events can be of two kinds:

- **Standard events** Events defined internally in Open Server.
- **Programmer-defined events** Events defined by the Open Client application. Adaptive Server Anywhere does not support any programmer-defined events.

Standard event handling

The following list describes each Open Server standard event, and the Adaptive Server Anywhere handling of that event.

- **SRV_ATTENTION event** This usually occurs when a client calls `ct_cancel` to stop results processing prematurely. Adaptive Server Anywhere checks the status of SRV_ATTENTION between fetches, and stops processing if the event has been received.
- **SRV_BULK event** A client has inserted a binary large object or has issued a bulk copy request. Inserting binary large objects is supported, but bulk copies are not supported by Adaptive Server Anywhere.
- **SRV_CONNECT event** A Client-Library client has called `ct_connect`. Connections to Adaptive Server Enterprise are made to the server, and the USE SQL statement sets the current database.
- **SRV_CURSOR event** A client has sent a cursor request. This event is supported by Adaptive Server Anywhere.
- **SRV_DISCONNECT event** A request to disconnect a client connection has been made. This event is supported by Adaptive Server Anywhere.
- **SRV_DYNAMIC event** A client has sent a Dynamic SQL request. Dynamic SQL allows a client application to execute SQL statements containing variables whose values are determined at run-time. All the operation types available for the SRV_DYNAMIC event are supported except for the DESCRIBE_INPUT type.

- **SRV_LANGUAGE event** A client has sent a language request, such as a SQL statement. The SQL statement is passed on to Adaptive Server Anywhere. The success of failure of the event depends on whether the SQL statement is supported by Adaptive Server Anywhere.

  For information about Transact-SQL compatibility in Adaptive Server Anywhere, see the chapter "Transact-SQL Compatibility" on page 781 of the book *Adaptive Server Anywhere User's Guide*.

- **SRV_MSG event** A client has sent a message. The message ID and parameters are read, but no action is performed. The success return code is returned to the client application.

- **SRV_OPTION event** A client has sent an option command. Options that are supported by Adaptive Server Anywhere are passed on to the database server.

- **SRV_RPC event** A client has issued a remote procedure call. The procedure is specified as follows:

  ```
  owner-name.database-name.procedure-name (parameter-
  list)
  ```

  Adaptive Server Anywhere checks to see if the current database is the correct one, and if so executes the procedure. If the current database is not the correct one, a failure is returned.

- **SRV_START event** A call to **srv_run** triggers a SRV_START event. This event is not generated by client applications.

- **SRV_STOP event** A request to stop Adaptive Server Anywhere has been made. This event is not generated by client applications. The **dbosstop** utility is provided for shutting down the Open Server.

- **SRV_URGDISCONNECT event** This event is only triggered by an Open Server application calling **srv_event**. It is not supported by Adaptive Server Anywhere.

Known limitations of Anywhere

Using the Open Client interface, you can use an Adaptive Server Anywhere database in much the same way as you would a SQL Server. There are some limitations, including the following:

- **Commit Service** Adaptive Server Anywhere does not support the SQL Server Commit Service.

- **Prepare Transaction statement** Adaptive Server Anywhere does not support the SQL Server Prepare Transaction statement used by DB-Library in a two-phase commit application to see if a server is prepared to commit a transaction.
- **Data types** Not all Adaptive Server Anywhere and SQL Server data types are supported in Adaptive Server Anywhere. For a description of data type compatibility, see "Data type mappings" on page 145.
- **Capabilities** A client/server connection's **capabilities** determine the types of client requests and server responses permitted for that connection. The following capabilities are not supported:
  - CS_REG_NOTIF
  - CS_CSR_ABS
  - CS_CSR_FIRST
  - CS_CSR_LAST
  - CS_CSR_PREV
  - CS_CSR_REL
  - CS_DATA_BOUNDARY
  - CS_DATA_SENSITIVITY
  - CS_PROTO_DYNPROC
  - CS_REQ_BCP

  ☞ For more information on capabilities, see the *Open Server Server-Library C Reference Manual*.