SYBASE®

User’s Guide for Access Services

Enterprise Connect Data Access Options

12.6

[Microsoft Windows and UNIX]

DOCUMENT ID: DC38454-01-1260-01

LAST REVISED: February 2005

Contents

About This Book

This book describes how to configure and use a DirectConnect™ access service, including datatype conversion, request processing, data transfer, and stored procedures.

Audience

This book is written for:

- Application Programmers, who develop organization-specific programs using the major features of DirectConnect.
- System Administrators, who install and test DirectConnect. When DirectConnect is running, System Administrators provide ongoing administration support, disaster recovery, and troubleshooting support.
- System Programmers, who install and test DirectConnect. System Programmers who provide product administration, troubleshooting, and disaster recovery.

How to use this book

This book covers the following topics:

Chapter	Topic
Chapter 1, “Introducing DirectConnect”	Introduces the product and describes the DirectConnect components.
Chapter 2, “Configuring the Access Service Library for DirectConnect”	Tells how to configure access service library and access service properties.
Chapter 3, “Configuring Access Services to Work with Related Products”	Provides instructions for setting up the Component Integration Services functionality in Adaptive Server® Enterprise (ASE/CIS) and Replication Server to use with a DirectConnect access service.
Chapter 4, “Querying and Setting Operating Values”	Explains how to use global variables and `set` statements to query and set operating values for your client connections.
Chapter 5, “Managing Transactions”	Describes the transaction management processing flow and explains how to configure properties to manage the process.
Chapter 6, “Issuing SQL Statements”	Describes SQL transformation modes and standard transformations for SQL commands.

Chapter	Topic
Chapter 7, “Issuing RPC Events”	Describes how to create, configure, and execute remote procedure calls (RPCs).
Chapter 8, “Understanding the Transfer Process”	Describes several concepts of the transfer process.
Chapter 9, “Using Bulk Copy Transfer and Express Transfer”	Describes how to use bulk copy transfer, including syntax statements and error handling.
Chapter 10, “Using Destination-Template Transfer”	Explains how to use destination-template transfer, including syntax statements and error handling.
Chapter 11, “Accessing Catalog Information with CSPs”	Provides a description and reference for supported catalog stored procedures (CSPs).
Chapter 12, “Retrieving Information with System Procedures”	Provides a description and reference for supported system procedures.
Appendix A, “Configuration Quick Reference”	Contains a quick reference table, in alphabetical order, for the configuration properties.
Appendix B, “Converting Datatypes”	Describes datatype conversions between ODBC and Open Server™.
Appendix C, “Using Stored Procedures”	Describes SQL stored procedures and DB2 stored procedures, including rules for using them.
Chapter D, “Using Sybase Mode Commands”	Describes SQL commands that the DirectConnect for ODBC access service recognizes in sybase mode.

Related documents

To configure and administer DirectConnect products, use the following guides:

- ECDA *Installation Guide* for Linux and UNIX
- ECDA *Installation Guide* for Windows
- ECDA and Mainframe Connect DirectConnect *Server Administration Guide* for DirectConnect

For Open Database Connectivity (ODBC) information, use the following document:

- Microsoft *ODBC 3.5 Programmer’s Reference and SDK Guide*

For additional references, use the following documents:

- Open Client™ and Open Server™ *Common Libraries Reference Manual*

- Open Client *Client-Library/C Reference Manual*
- Open Client *Client-Library/C Programmer's Guide*
- Open Server *Server-Library/C Reference Manual*
- Software Developer's Kit and Open Server *Installation Guide* for Microsoft Windows and UNIX
- Component Integration Services *User's Guide for Adaptive Server Enterprise and OmniConnect*
- Sybase® Adaptive Server *Reference Manual*, volumes 1 and 2

To configure and administer the DirectConnect server, use the DirectConnect *Server Administration Guide*.

Other sources of information

Use the Sybase Getting Started CD, the Sybase Technical Library CD, and the Technical Library Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the Technical Library CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader (downloadable at no charge from the Adobe Web site, using a link provided on the CD).
- The Technical Library CD contains product manuals and is included with your software. The DynaText reader (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.
- The Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Technical Library Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.
5. Click a Certification Report title to display the report.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1. Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.
2. Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
3. Select a product.
4. Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

   Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.
5. Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Style conventions

This book uses the following style conventions:

- The names of files and directories are shown as:

  econnect\ServerName\CFG

- The names of programs, utilities, procedures, and commands are shown as:

  the set statement

- The names of properties are shown as:

  Allocate

- The names of options are shown as:

  connect

- Code examples and text on screen are shown as:

  ** Prepare the statement.

- In a sample command line display, commands you should enter are shown as:

  ```
  Allocate=connect
  ```

- In a sample command line display, variables (which are words you should replace with the appropriate value for your system) are shown as:

  ```
  ClientIdleTimeout=integer
  ```

Syntax conventions

Syntax statements that display options for a command look like this:

sp_columns *table_name* [, *table_owner*]
 [, *table_qualifier*] [, *column_name*]

The following table explains the syntax conventions used in this book.

Table 1: Syntax conventions

Symbol	Convention
( )	Include parentheses as part of the command.
{ }	Braces indicate that you must choose at least one of the enclosed options. Do not type the braces when you type the option.
[ ]	Brackets indicate that you can choose one or more of the enclosed options, or none. Do not type the brackets when you type the options.
|	The vertical bar indicates that you can select only one of the options shown. Do not type the bar in your command.
,	The comma indicates that you can choose one or more of the options shown. Separate each choice by using a comma as part of the command.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1 Introducing DirectConnect

Before Enterprise Connect Data Access 12.5, its options were sold as individual DirectConnect products named “DirectConnect for [target].” Other than in the installer, you will see the name “DirectConnect” used in the software and in documents. As such, this document uses the old product name except for the title page. Old and current option names are as follows:

Table 1-1: DirectConnect and ECDA naming

Old product name	Current option name
DirectConnect for DB2 UDB	Enterprise Connect Data Access Option for DB2 UDB
DirectConnect for Informix	Enterprise Connect Data Access Option for Informix
DirectConnect Manager	Part of each Enterprise Connect Data Access Option
DirectConnect for Microsoft SQL Server	Enterprise Connect Data Access Option for Microsoft SQL Server
DirectConnect for ODBC	Enterprise Connect Data Access Option for ODBC
DirectConnect for Oracle	Enterprise Connect Data Access Option for Oracle

For more information, see the Enterprise Connect Data Access *Overview Guide*.

This chapter introduces basic DirectConnect concepts and describes the process that provides access to distributed data.

It contains the following topics:

For more information, see the Enterprise Connect Data Access *Overview Guide*.

DirectConnect overview

This section describes DirectConnect and other Sybase products that DirectConnect and access service interact with. This section covers the following topics:

- DirectConnect architecture
- DirectConnect components

DirectConnect architecture

DirectConnect is Open Server™-based software that supports Client-Library,™ and Open Database Connectivity (ODBC) application programming interfaces (APIs).

Note Applications built on the Open Server API depend on a specific directory structure environment and routing tables. For an explanation of the installed directory structure, see the "Directory structure" section in the specific chapter for your DirectConnect product.

DirectConnect serves as a fundamental building block for highly-scalable database middleware applications. DirectConnect products are local area network (LAN)-based middleware servers that provide access to non-Sybase data and applications.

In addition, DirectConnect can be used with other Sybase products, such as Adaptive Server Enterprise/Component Integrated Services (ASE/CIS), Sybase Adaptive Server, and Replication Server.

DirectConnect consists of:

- A server, which provides the framework in which service libraries can operate
- One or more service libraries, which provide the framework in which access services can operate
- One or more access services per service library, which are the logical points of connection for DirectConnect clients

The following subsections describe each of these components.

DirectConnect components

This section describes the following DirectConnect components:

- DirectConnect server
- DirectConnect service libraries
- DirectConnect access services

DirectConnect server

The DirectConnect server provides management and support functions for DirectConnect service libraries, such as:

- Routing client connections to the appropriate access service based on user ID, requesting application, and access service name
- Providing a single log file and a trace file for access services
- Logging server, access service, and client messages
- Tracing server, access service, and client events
- Providing configuration management of all installed services

For detailed information about configuring and starting the server, see the DirectConnect *Server Administration Guide*.

DirectConnect service libraries

The following service libraries reside on the DirectConnect server:

- Access service library, a set of configuration properties that describes how all of its access services will function
- Administrative service library, which provides specific administrative services for all DirectConnect libraries, including writing to logs and allowing remote configuration of DirectConnect access services (for example, through DirectConnect Manager).

DirectConnect access services

An access service is the client connection point for a DirectConnect server. You can think of it as the pairing of a service library with a set of specific values for the configuration properties. You must define at least one access service for every service library.

Access services

Access services allow clients to access data from a specific target. Each access service is a specific set of configuration properties that:

- Transforms SQL
- Convert datatypes
- Supports remote procedure calls (RPCs)
- Transfers data between the target database and other servers accessible through Open Client
- Supports Catalog Stored Procedures (CSPs) and system stored procedures

DirectConnect Manager

DirectConnect Manager is a graphical user interface (GUI) systems management tool for administering DirectConnect. DirectConnect Manager runs on Windows and UNIX platforms, and provides the following capabilities:

- Manage DirectConnect servers on multiple platforms.
- Change configuration properties of DirectConnect servers, service libraries, and services.
- Create and copy services by copying an existing service and giving it a unique name.
- Create new servers.
- Start and stop existing servers.
- Start, stop, and delete services (from a remote site, DirectConnect Manager is the only way you can start a service).
- Test the availability of a data source by creating a connection to it.
- Retrieve a DirectConnect server log file or a subset of the log, and view log file messages with a text editor.
- Update DirectConnect server connection information.
- View the status of a service and data source on the desktop.

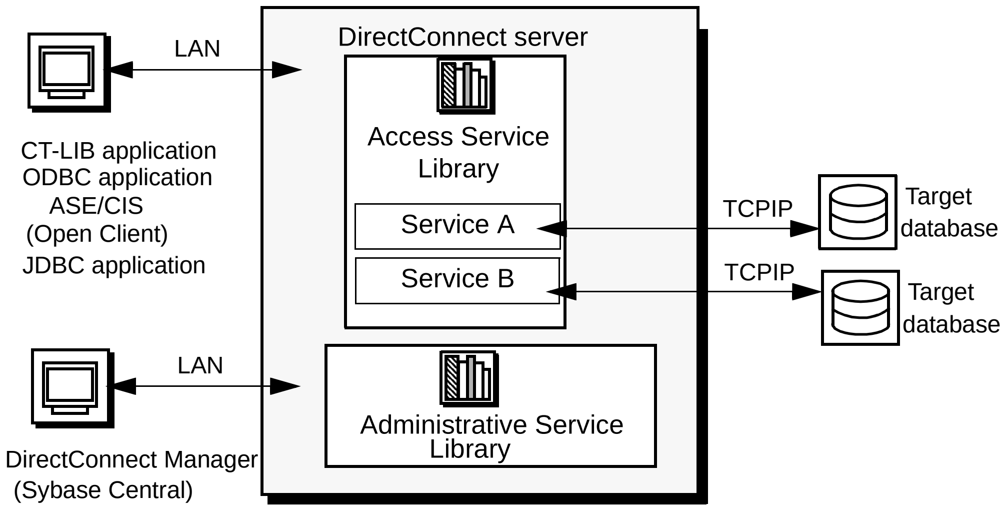
DirectConnect environment

DirectConnect consists of a server and one or more service libraries. The server provides the framework in which the service libraries operate. Each access service library accesses data from a particular target database, such as DB2 UDB, Informix, Microsoft SQL Server, and any ODBC-accessible database. It consists of one or more access services that have specific sets of configuration properties, as shown in Figure 1-1.

When a client connects, the access service logs in to the target database, using the client user ID and password, plus the ODBC-configured data source name (DSN).

The following figure shows the relationship of the access service library with components of the client workstation, LAN, ODBC driver, and target database.

Figure 1-1: DirectConnect environment



As shown, the request from a client application goes over the LAN to the DirectConnect server. The DirectConnect access service routes the request through the ODBC driver to the target and accesses data from the target database.

DirectConnect data access products

Following are descriptions of the four DirectConnect data access products that are covered in this guide. In particular, they provides access management, copy management, and remote systems management. Each access service product contains one or more access services that contain specific sets of configuration properties relating to the target to be accessed:

- DirectConnect for DB2 UDB
- DirectConnect for Informix
- DirectConnect for Microsoft SQL Server
- DirectConnect for ODBC

Note For configuration properties for DirectConnect for Oracle, see the DirectConnect for Oracle *User's Guide*.

DirectConnect for DB2 UDB

DirectConnect for DB2 UDB provides access to DB2 UDB databases using a DRDA protocol. A DRDA-to-ODBC driver is supplied and used to access the DB2 UDB target databases.

DRDA, or Distributed Relational Database Architecture, is IBM's wire protocol for access to DB2 UDB. DirectConnect for DRDA supports access to DB2 UDB on z/OS, AS/400, Windows, and UNIX platforms.

DirectConnect for DB2 UDB consists of a DirectConnect server, DirectConnect service libraries, and DirectConnect access services. The DirectConnect for DB2 UDB access service library accesses DB2 UDB databases.

DirectConnect for Informix

DirectConnect for Informix provides basic connectivity to Informix data sources. DirectConnect for Informix consists of a DirectConnect server, DirectConnect for Informix service libraries, and DirectConnect for Informix access services.

DirectConnect for Microsoft SQL Server

DirectConnect for Microsoft SQL Server provides basic connectivity to Microsoft SQL Server data sources. It consists of a DirectConnect server, DirectConnect for Microsoft SQL Server service libraries, and DirectConnect for Microsoft SQL Server access services.

DirectConnect for ODBC

DirectConnect ODBC provides basic connectivity to non-Sybase data sources. DirectConnect for ODBC consists of a DirectConnect server, DirectConnect for ODBC service libraries, and DirectConnect for ODBC access services.

The DirectConnect for ODBC access service library accesses ODBC-accessible data. Each access service library accesses data from a particular target database, such as DB2 UDB, Informix, and other ODBC-accessible databases.

Note The ODBC driver for DirectConnect for ODBC is not provided by Sybase; you must select your driver, install it, and maintain it.

DirectConnect for DB2 UDB, DirectConnect for Microsoft SQL Server, and DirectConnect for Informix consist of the base DirectConnect for ODBC component, combined with ODBC drivers supplied by Sybase. For access to additional databases such as Microsoft Access, Teradata, and so on, you must obtain and install the necessary ODBC driver separately, on the same server as DirectConnect for ODBC, and configure DirectConnect for ODBC to use that ODBC driver for access to the database. Since ODBC drivers have varying degrees of functionality, it is important when working with non-Sybase-provided, third-party ODBC drivers to carefully integrate and test them to be sure they meet your needs

DirectConnect for Oracle

DirectConnect for Oracle is a Sybase product that provides access to Oracle databases.

The initial installation of Oracle is described in Chapter 6, “Installing DirectConnect for Oracle 12.6.” However, this product is discussed in detail in the ECDA Option for Oracle *Server Administration and User’s Guide*.

Globalization

Globalization consists of internationalization and localization of messages.

Internationalization

Internationalization consists of character code set conversion and cultural formatting.

Code set conversion involves converting the hexadecimal representation of a character from a code set in a target database to a code set in a client application, or the reverse.

Cultural formatting involves designating decimal separators, monetary signs, date and time separators, and a three-digit grouping symbol. Cultural formatting in DirectConnect is performed through the use of configuration properties.

Supported date and time formats

To facilitate internationalization, the access service can be set to return dates and time in any one of the following formats:

- char_iso: *yyyy-mm-dd-hh.mm.ss.nnnnnn*
- char_usa: *mm/dd/yyyy hh:mm AM or PM*
- char_eur: *dd.mm.yyyy hh.mm.ss*
- char_jis: *yyyy-mm-dd hh:mm:ss*
- char_odbc: *yyyy-mm-dd hh:mm:ss.nnnnnn*

Code page translation

For Open DataBase Connectivity (ODBC)-based products, code page translation can take place in two locations:

- Between DirectConnect and the target database
- Between the client and DirectConnect

For additional information regarding code page translation refer to Chapter 2, “Configuring the Access Service Library for DirectConnect.”

Localization

Two sets of messages can be localized:

- Messages generated by the target database manager and passed to the client application without change

  The target database manager can be any application between DirectConnect and the target data file, including the ODBC driver.

- Messages generated in DirectConnect

  DirectConnect does not localize database manager messages. For information on how to set up localization of such messages, see your database manager and the ODBC driver documentation.

  Messages generated in DirectConnect are always localized.

.

CHAPTER 2

Configuring the Access Service Library for DirectConnect

This chapter describes how to configure properties to customize the access service library and the individual access services for the following DirectConnect products:

- DirectConnect for DB2 UDB
- DirectConnect for Informix
- DirectConnect for Microsoft SQL Server
- DirectConnect for ODBC

Note For the configuration properties for DirectConnect for Oracle, refer to the Enterprise Connect Data Access Option for Oracle *Server Administration and User's Guide*.

This chapter covers the following topics:

For an alphabetized list of all configuration properties, see Table A-1, Appendix A, “Configuration Quick Reference.”

Understanding the configuration process

To create additional access services and to edit, configure, and change existing properties in the access service library configuration file, use one of the following:

- Use DirectConnect Manager to modify the access service library configuration file and dynamically change the properties without stopping and starting the server.
- Use a line text editor to edit the access service library configuration file that resides on the DirectConnect server. Upon completion, you must stop and restart the server for the changes to take effect.

Note For convenience and ease of use, Sybase recommends that you use DirectConnect Manager to modify the access service configuration file.

For information about using DirectConnect Manager, see the DirectConnect Manager online help.

The access service library uses some configuration information from the DirectConnect server which can be found in the ECDA and Mainframe Connect *Server Administration Guide* for DirectConnect.

Description of the configuration file

Use DirectConnect Manager or your text editor to modify and save the configuration file named *dcany.cfg*. Configuration files are defined in this section for:

- DirectConnect for DB2 UDB
- DirectConnect for Informix
- DirectConnect for Microsoft SQL Server
- DirectConnect for ODBC

To find the location of the configuration file within the DirectConnect directory structure, see the DirectConnect installation guides, version 12.6, for Windows and UNIX.

Each access service has a specific set of configuration properties. To configure an access service, perform the following tasks:

- Enter site-specific values for all required properties.
- For non-required properties, enter only the values that differ from the default values.

The following principles apply to properties:

- Server properties apply to all access services created for that server.
- Service library properties apply to the service library.
- Access service properties apply to specific access services.

Note Configuration properties are not case sensitive.

Configuration file format

An access service library configuration file consists of the following:

- A primary section [*Service Library*] that groups access service library properties. The name is hard-coded and cannot be changed.
- Access service sections [*Service Name*], shown in brackets.
- Subsections {*Subsection Name*}, shown in braces. Subsections group the properties by type.
- Configuration properties and values.

You can include comments. Enter each comment on a separate line and begin with a semicolon or the "#" symbol in column one.

Configuration templates

The following templates for DirectConnect targets show all of the DirectConnect configuration properties. Use these templates to set up your configuration file.

Configuration template for DirectConnect

In the following template, there are two required properties indicated with an asterisk (*), that have entries for DirectConnect for Microsoft SQL Server. For guidelines on configuring these properties, see “List of configuration property categories” on page 20.

```
[Service Library]
{ACS Required}
*ConnectionSpec1=dcmsss

{Catalog Stored Procedures}
CSPColumnODBCVersion=
CSPExclusions=
CSPIncludeAlias=
CSPIncludeSynonym=
CSPIncludeSystem=
CSPIncludeTable=
CSPIncludeView=
DatatypeInfo=

{Client Interaction}
ClientDecimalSeparator=
ClientIdleTimeout=
*EnableAtStartup=yes
MaxResultSize=
MaxRowsReturned=
MaxSvcConnections=
quoted_identifiers=
SendWarningMessages=
ServiceDescription=
StripBinaryZero=
SvclibDescription=
TextSize=
TransactionMode=
Version=

{Data Conversion Errors}
CharConvertError=
DateTimeConvertError=
DefaultDate=
DefaultNum=
DefaultTime=
NumConvertError=

{Datatype Conversion}
BinaryResults=
DateResults=
DateTimeResults=
DecimalResults=
FloatResults=
Int2Results=
Int4results=
RealResults=
TimeResults=
TinyIntResults
XNLChar=
XNLVarChar=

{Logging}
LogConnectionStatistics=
LogReceivedSQL=
LogRequestStatistics=
LogServiceStatistics=
LogSvclibStatistics=
LogTargetActivity=
LogTransferStatistics=
LogTransformedSQL=

{Target Interaction}
Allocate=
DelimitSQLRequests=
IsolationLevel=
QuotedStringDelimeter=
ReturnNativeError=
SQLOdbcCursors=
SQLTransformation=
StopCondition=
TargetDBMS=
TargetDecimalSeparator=

{Tracing}
TraceEvents=
TraceInterface=
TraceTarget=

{Transfer}
BulkCommitCount=
TransferBatch=
TransferErrorAction
TransferErrorCount=
TransferExpress=
```

How to change configuration property values

Although most access service configuration property values have default values, you will need to change some configuration property values for your site. You can change property values either by using DirectConnect Manager or by editing the text file.

Using DirectConnect Manager

For instructions on how to use DirectConnect Manager to edit the access service configuration file (*dcany.cfg*), go to the Managing Access Services topic of DirectConnect Manager online Help and select “Modifying access service configuration properties.”

Note Before you can use DirectConnect Manager to update the access service properties, you must have installed DirectConnect Manager as outlined in the Installation Guide for your platform, and you must also identify and establish a connection between the server and DirectConnect Manager. This is described in DirectConnect Manager online Help topic, “Connecting DirectConnect Manager to a DirectConnect Server.”

For additional information, refer to DirectConnect Manager online help or use the verbose mode that is available with DirectConnect Manager.

Using the text editor

To edit the configuration property values using the text editor:

1. Open the Access Service Library configuration file. Update the service library configuration properties.
2. Open the access services file and change the access service property values as applicable.

   List each property and value under the appropriate subsection. If the subsection is not shown, you must add it.
3. Save the file.
4. Stop the server, and then restart it to implement the changes.

How to create additional services

You can create additional access services in the same way you change existing ones, either by using the text editor to edit the text file or by using DirectConnect Manager.

Using DirectConnect Manager

For instructions on how to use DirectConnect Manager to create a service, go to the Managing Access Services topic of the DirectConnect Manager online Help and select “Creating a new service” or “Copying a service.”

Using the text editor

To create a service or additional services:

1. Open the Access Service Library configuration file called *dcany.cfg*.
2. Create a section for each new service and add:
   - The service name, in brackets
   - Required properties below the service name, grouped in the {ACS Required} subsection
   - Property value overrides listed below the service name, grouped by subsection
3. Save the file.
4. Stop the server, then restart it to implement the changes.
5. To connect to a new access service from a client machine, you must enter the access service name in the *sql.ini* configuration file on Windows NT machines or the *interfaces* file on UNIX client machines. If you choose to use service name redirection, make an assigned service name entry in the service name redirection file.

For instructions about editing the *sql.ini* or *interfaces* file, see the ECDA *Installation Guide* for UNIX and the ECDA *Installation Guide* for Microsoft Windows.

For information about service name redirection, see the ECDA and Mainframe Connect *Server Administration Guide* for DirectConnect.

Guidelines for access service names

Access service names must conform to the following rules:

- Service names must be unique (without regard to case).
- Service names must not exceed 31 characters for Windows or UNIX operating systems.
- The initial character must be an alphabetic character (a–z, A–Z); subsequent characters can be alphabetic characters, numbers, or the underscore (_) character.

Code page translation for ODBC-based products

The ODBC drivers have been updated to incorporate code page translation within their normal data processing. DirectConnect uses this new functionality to simplify code page translation.

For Open DataBase Connectivity (ODBC)-based products, code page translation can take place in two locations:

- Between DirectConnect and the target database
- Between the client and DirectConnect

The ODBC drivers have been updated to incorporate code page translation within the driver’s normal data processing. DirectConnect uses this functionality to simplify code page translation.

Between DirectConnect and the target database

The ODBC driver uses the server-platform-configured code page value as its client code page. Depending on the platform, Windows or UNIX, the server-platform-configured code page value can be found as follows:

- For Windows, in the Windows registry ACP value. To locate the value, use the registry editor called regedit to navigate through the registry tree to *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\NIs\CodePage*. On the right panel, scroll to the ACP value.
- For UNIX, it is the IANAAppCodePage property value.

DirectConnect is an Open Server API and relies on Open Server for datatype conversion. Upon connection to the target, the ODBC driver queries the target database for its code page and compares the value to the server-platform-configured code page. If the values are not equal, the ODBC driver translates from the server-platform-code page to the target code page. If the values are equal, the ODBC driver does not perform any translation.

Between the client and DirectConnect

For proper code page translation, the DirectConnect code page identified by Open Server must match the server-platform-configured code page value. The *locales.dat* platform configuration identifies the Open Server code page. Configure the *locales.dat* default for the appropriate platform. Open Server handles code page conversion as follows:

- If the client code page matches the Open Server code page no conversion is performed by Open Server.
- If the client code page does not match the Open Server code page, Open Server performs the conversion.

The DirectConnect server configuration property called OSCodeSetConvert determines whether DirectConnect enables Open Server to perform code page translation between the client and the ODBC driver. The following are the two values for the new OSCodeSetConvert property:

- Yes indicates that DirectConnect performs code page translation.
- No indicates that DirectConnect will not perform any code page translation.

Access service properties no longer available

With the introduction of the new server configuration property OSCodeSetConvert and the new functionality of the ODBC driver, the following access service properties are no longer available:

- UseClientCharset
- DefaultClientCodeset
- DefaultTargetCodeset

For additional information for the OSCodeSetConvert property, refer to the ECDA and Mainframe Connect *Server Administration Guide* for DirectConnect.

List of configuration property categories

The following sections describe by property category; property, syntax, range, default values, acceptable values, and comments:

- ACS required properties
- Catalog stored procedures properties
- Client interaction properties
- Data conversion error properties
- Datatype conversion properties
- Logging properties
- Target interaction properties
- Tracing properties
- Transfer properties

See Appendix A, "Configuration Quick Reference," for an alphabetized listing of all configuration properties within the Access Service Library for DirectConnect.

Note Some properties require specific values for your site. These required properties reside in the access service required property category. For information, see ACS required properties, in the following section.

ACS required properties

These properties require site-specific values. Be sure to supply these values for your installation.

The subsection heading and a list of the properties must appear in the access service library configuration file as shown:

```
{ACS Required}
ConnectionSpec1=
```

ConnectionSpec1

Specifies an ODBC data source name (DSN) defined in the *ODBC system information* file.

In UNIX systems, the data sources are defined in an *odbc.ini* file. In Windows NT systems, the data sources are defined using the ODBC Administrator.

Syntax	ConnectionSpec1=*char*
Range	1–255 characters
Default	None
Value	*char* is a valid data source name configured in the *ODBC system information* file.

Catalog stored procedures properties

These properties control the information an access service returns from catalog stored procedures.

Many of the properties in this group are not supported, nor do they affect the DirectConnect access service. These properties are available for compatibility purposes only.

The subsection heading and a list of the properties must appear in the access service library configuration file as shown:

```
{Catalog Stored Procedures}

CSPColumnODBCVersion=
CSPExclusions=
CSPIncludeAlias=
CSPIncludeSynonym=
CSPIncludeSystem=
CSPIncludeTable=
CSPIncludeView=
DatatypeInfo=
```

CSPColumnODBCVersion

Specifies ODBC version that catalog stored procedures results conform to. This affects interoperability with ASE/CIS.

Syntax	CSPColumnODBC Version = [2 | 3]
Default	3
Values	• 2 specifies ASE/CIS version 12.0. • 3 specifies ASE/CIS version 12.5 and later.

Comment	This property affects interoperability with ASE/CIS. In the event that the following error occurs:

```
Error 11209 - ‘column type mismatch in remote object’
when executing create existing table command to MSSQL
server 2000SP 2.
```

Change the value from the default value to 2.

CSPExclusions

Specifies an access service to limit access to information normally returned from sp_tables upon authorization.

Syntax

```
CSPExclusions=[ none | user | nonauth |nonauthpublic ]
```

Default: user

Values

- none specifies no exclusions, based upon authorization to information.
- user specifies exclusions, based upon specific user authorization to information.
- nonauth specifies that a user must be granted user or group authorization to access information.
- nonauthpublic specifies that a user must be granted user authorization, or PUBLIC is granted some authorization.

Comment: This property is not supported and does not affect the DirectConnect access service. It is available for compatibility purposes only.

CSPIncludeAlias

Specifies the access service to return information about aliases from sp_tables.

Syntax

```
CSPIncludeAlias=[no | yes]
```

Default: no

Values

- no specifies that the access service does not return alias information.
- yes specifies that the access service returns alias information.

Comment: This property is not supported and does not affect the DirectConnect access service. It is available for compatibility purposes only.

CSPIncludeSynonym

Specifies the access service to return information about synonyms from sp_tables.

Syntax	CSPIncludeSynonym=[no | yes]
Default	no
Values	• no specifies the access service not to return synonym information. • yes specifies the access service to return synonym information.

CSPIncludeSystem

Specifies the access service to return information about system tables from sp_tables.

Syntax	CSPIncludeSystem=[no | yes]
Default	no
Values	• no specifies the access service not to return system table information. • yes specifies the access service to return system table information.
Comment	The user issuing sp_tables must be authorized to query these tables.

CSPIncludeTable

Specifies the access service to return information about tables from sp_tables.

Syntax	CSPIncludeTable=[yes | no]
Default	yes
Values	• yes specifies the access service to return table information. • no specifies the access service not to return table information.

CSPIncludeView

Specifies the access service to return information about views from sp_tables.

Syntax	CSPIncludeView=[yes | no]
Default	yes
Values	• yes specifies the access service to return view information. • no specifies the access service not to return view information.

DatatypeInfo

Specifies the type of datatype information returned from sp_datatype_info.

Syntax

    DatatypeInfo=[transact | target]

Default

transact

Values

- transact specifies sp_datatype_info to return the Transact-SQL™ datatypes that map to the ODBC datatypes supported by the target.
- target specifies sp_datatype_info to return target datatype names.

Client interaction properties

These properties control how an access service library or an access service interacts with client applications.

The subsection heading and a list of the properties must appear in the access service library configuration file as shown:

```
{Client Interaction}
ClientDecimalSeparator=
ClientIdleTimeout=
EnableAtStartup=
MaxResultSize=
MaxRowsReturned=
MaxSvcConnections=
quoted_identifier=
SendWarningMessages=
ServiceDescription=
StripBinaryZero=
SvclibDescription=
TextSize=
TransactionMode=
Version=
```

ClientDecimalSeparator

Specifies the character the client application uses to separate decimal numbers for presentation purposes.

Syntax

ClientDecimalSeparator=*char*

Default

. (period)

Values

- *char* indicates the character used by the client application as the decimal delimiter.

- A period indicates that the client application uses a period as the decimal delimiter.

Comment	If your client application uses a different character as a decimal delimiter, verify that this application connects to an access service configuration set that uses the same client decimal delimiter character.

ClientIdleTimeout

Specifies how many minutes a client connection can remain inactive before an access service terminates the connection.

Syntax

```
ClientIdleTimeout=integer
```

Range: 0–1024

Default: 0

Values

- *integer* is the number of minutes a client connection can remain inactive before an access service terminates the connection.
- 0 indicates that an access service never terminates an idle connection.

Comments

- A connection is idle when any of the following occurs:
  - A client application connects but does not issue a command.
  - A command completes processing, but the client does not issue a new command.
  - A large result set is returned from a SQL request, and the result screen paused for the specified timeout period.
- The access service checks client activity once per minute. Therefore, a ClientIdleTimeout value of n might allow a client to remain active for nearly n + 1 minutes.

EnableAtStartup

Specifies whether this access service starts when the DirectConnect server starts.

Syntax

```
EnableAtStartup=[no | yes]
```

Default: no

Values

- no means that the access service does not start when the server starts.
- yes means that the access service starts when the server starts.

Comment	If you are not using DirectConnect Manager, set this property to yes.

MaxResultSize

Specifies the maximum number of bytes an access service returns to the client application in a result set.

Syntax	MaxResultSize=*integer*
Range	0–2147483646
Default	2147483646
Values	• *integer* is a number of bytes. • A value of 0 defaults to 2147483646.
Comments	• The MaxResultSize value is approximate in that the access service checks at the end of each row to see if the value is exceeded. • If the value is exceeded, the access service does the following: • Sends the entire row to the client application (not a partial row) • Does not send any of the remaining rows in the result set

MaxRowsReturned

Specifies the maximum number of rows an access service returns to the client application in a result set.

Syntax	MaxRowsReturned=*integer*
Range	0–2147483646
Default	2147483646
Values	• *integer* is a number of rows. • A value of 0 defaults to 2147483646.
Comment	If the MaxRowsReturned value is exceeded, the value is returned to the client as an error.

MaxSvcConnections

Specifies the maximum number of client connections that can log onto the access service at one time.

Syntax	MaxSvcConnections=*integer*

Range	1–*n*, where *n* is the maximum number of client connections allowed for the access service.
Default	MaxConnections property value of the DirectConnect server
Value	*integer* is a number of client connections.
Comments	• If you set the Allocate property value to request, this allows more actual clients to be supported because unused clients will not be counted as connections. • The DirectConnect server MaxConnections property determines the maximum number of client connections. For information about MaxConnections, see the ECDA and Mainframe Connect *Server Administration Guide* for DirectConnect.

quoted_identifier

Specifies whether to enable or disable delimited identifiers. Delimited identifiers are object names enclosed in double quotes. You can use them to avoid certain restrictions on object names. Table, view, and column names can be delimited by quotes; other names cannot.

Delimited identifiers can:

- Be reserved words
- Begin with non-alphabetic characters
- Include characters that ordinarily are not allowed

Syntax

quoted_identifer=[on | off]

Default

off

Values

- on means that a quoted string used as an identifier is recognized.
- off means that a quoted string used as an identifier is not recognized as an identifier.

Comments

- Delimited identifiers follow the conventions for identifiers for DB2.
- Before you create or reference a delimited identifier, issue the following statement:

  ```
  set quoted_identifier on
  ```

  Each time you use the delimited identifier in a statement, you must enclose it in double quotes. For example:

  ```
  create table “1one” (col 1 char(3))
  ```

```
 create table “include spaces” (col1 int)
```

or

```
create table “grant” (“add” int)
 insert into “grant” (“add”) values (3)
```

- When the quoted_identifier configuration property is turned *on*, use single quotes, not double quotes, around character or date strings. Delimiting strings with double quotes causes Adaptive Server to treat them as identifiers. The following example shows the correct way to insert a character string into col1 of 1one when the quoted identifier “1one” is turned on:

```
insert into “1one”(col1) values (‘abc’)
```

- To insert a single quote into a column, use two consecutive single quotation marks. The following example shows the correct way to insert the values “a’b” into col1:

```
insert “1one”(col1) values(‘a”b’)
```

SendWarningMessages

Specifies whether an access service returns warning messages to the client application.

Syntax	SendWarningMessages=[no | yes]
Default	no
Values	• no specifies the access service not to return warning messages to the client application. • yes specifies the access service to return warning messages to the client application.

ServiceDescription

Allows you to place descriptive information about each access service in the configuration file.

Syntax	ServiceDescription=*char*
Range	0–255 characters
Default	None
Value	char is a user-defined character string.

StripBinaryZero

Specifies whether binary zeros are removed from the incoming language commands.

Syntax	StripBinaryZero=[yes | no]
Default	no
Values	• no specifies the access service not to remove binary zeros from the incoming language commands. • yes specifies the access service to remove binary zeros from the incoming language commands.

SvclibDescription

Describes the access service library. This property applies to a description of the access service library.

Syntax	SvcLibDescription=*char*
Range	0–255 characters
Default	None
Comment	This property allows you to place descriptive information about the access service library in the configuration file.

TextSize

Specifies the maximum number of bytes in character columns an access service returns to the client application.

Syntax	TextSize=*integer*
Range	0 - 2147483647
Default	2147483647
Values	• *integer* is a number of bytes. • A value of *0* defaults to *2147483647*.
Comments	• The access service truncates data exceeding the TextSize length and does not issue a warning message. • The access service returns character data longer than XNLCHAR or XNLVARCHAR bytes as CS_TEXT.

- For DirectConnect for Microsoft SQL Server, how the data is queried determines the text and image results processing. Queries with a select list of a single text or image column results in data streaming. If data is streamed, a maximum of 2,147,483,647 bytes may be returned per text and image value. Queries with a select list containing more than one column results in bound data. All bound data, including text and image is limited to 32,767 bytes. The textsize configuration property applies to both streaming and bound character data.

Note Text data manipulation using text pointers is only supported for DirectConnect for Microsoft SQL Server.

TransactionMode

Specifies whether the access service or the client application manages commit and rollback statements.

Syntax

```
TransactionMode=[short | long]
```

Default

short

Values

- long specifies the access service to give commitment control to the client application.
- short specifies the access service to issue a commit or a rollback after each request.

Comments

- The access service holds open the connection to the data source until the client application issues a commit or rollback, or until the ClientIdleTimeout value is exceeded.
- If ClientIdleTimeout is exceeded, the transaction rolls back.

Version

Specifies the version string for DirectConnect.

Syntax

Version=*versionstring*

Default

The access service default version string

Value

versionstring is the version string that is reported to client applications.

Comments

- This property allows you to customize a version string for client applications that rely on a string that is different from the access service default.

- If you customize an alternate version string, the following rules apply:
  - The string format cannot contain embedded new lines.
  - You can insert a space after the equal sign for readability in the configuration file. However, when an access service sends the version string to the client application, it removes any leading and trailing white space.
- You can obtain the access service default version string by issuing sp_helpserver (see sp_helpserver on page 150).

Data conversion error properties

These properties control the action an access service takes when it encounters data conversion errors.

The subsection heading and a list of the properties must appear in the access service library configuration file as shown:

```
{Data Conversion Errors}
CharConvertError=
DateTimeConvertError=
DefaultDate=
DefaultNum=
DefaultTime=
NumConvertError=
```

CharConvertError

Specifies the action an access service takes when it encounters a results column that is too long for the target column.

Syntax

CharConvertError=[reject | truncate]

Default

reject

Values

- reject specifies the access service to reject the row containing the error and issues a warning message.
- truncate specifies the access service to insert data to the length of the target column, truncate the remaining data, and issue a warning message.

DateTimeConvertError

Specifies the action an access service takes when it encounters rows with date, time, or datetime data values that are out of range for the target datatype.

Syntax	`DateTimeConvertError=[reject	null	default]`
Default	reject

Values

- reject specifies the access service to reject the row containing the error and issues a warning message.
- null specifies the access service to insert a NULL into the column and issues a warning message.
- default specifies the access service to insert the default date and time values, as configured in the DefaultDate and DefaultTime properties, into the column and issue a warning message.

DefaultDate

Specifies the value an access service inserts into columns with date conversion errors when DateTimeConvertError is set to default.

Syntax	DefaultDate=*yyyy-mm-dd*
Default	1900-01-01

Value

yyyy-mm-dd is the default, where:

- *yyyy* is the year.
- *mm* is the month.
- *dd* is the day.

DefaultNum

Specifies the value an access service inserts into columns with numeric conversion errors when NumConvertError is set to default.

Syntax	DefaultNum=*integer*
Default	0

Values

integer is a valid number that replaces the value that caused the conversion error.

DefaultTime

Specifies the value an access service inserts into columns with time conversion errors when DateTimeConvertError is set to default.

Syntax

    DefaultTime=*hh.mm.ss*

Default

00.00.00

Value

hh.mm.ss is the default, where:

- *hh* is the hour in 24-hour clock time.
- *mm* is the minute.
- *ss* is the second.
- A period is used as the delimiter.

NumConvertError

Specifies the action an access service takes when it encounters rows with numeric data values that are out of range for the target datatype.

Syntax

    NumConvertError=[reject | null | default]

Default

reject

Values

- reject specifies the access service to reject the row containing the error and issues a warning message.
- null specifies the access service to insert a NULL into the column and issues a warning message.
- default specifies the access service to insert the default numeric value configured in the DefaultNum property into the column and issue a warning message.

Datatype conversion properties

These properties control how an access service converts target database datatypes to Open Client and Open Server datatypes before sending the data to the client application.

The subsection heading and a list of the properties must appear in the access service library configuration file as shown:

```
{Datatype Conversion}
BinaryResults=
DateResults=
DateTimeResults=
DecimalResults=
FloatResults=
Int2Results=
Int4results=
RealResults=
TimeResults=
TinyIntResults=
XNLChar=
XNLVarChar=
```

To provide portability across the DBMS, the names of the configuration properties refer to generic datatypes. The description includes specific target database datatypes to which these generic datatypes correspond.

Note Datatype conversion properties control conversion of outgoing data from the DBMS. These properties do not control conversion of incoming data from client applications.

For information about converting incoming data from client applications, see Appendix A, "Configuration Quick Reference."

BinaryResults

Specifies the Open Server datatype to which returned binary results are converted.

Syntax

BinaryResults=[binary | char]

Default

binary

Values

- binary indicates that results of 255 bytes or less are returned as CS_BINARY. Those with 256 bytes or more are returned as CS_IMAGE.
- char indicates that results of 255 bytes or less are returned as CS_CHAR. Those with 256 bytes or more are returned as CS_TEXT.

Comments

If the BinaryResult configuration property is set to char, the access service changes the binary string to a character string and converts the character string to the client code set.

DateResults

Specifies the Open Client and Open Server datatype to which an access service converts DATE results.

Syntax

```
DateResults=[datetime | datetime4 | char_iso | char_usa | char_eur |
char_jis | char_odbc]
```

Default

datetime

Values

- datetime returns an 8-byte datetime datatype with a range of legal values from January 1, 1753, to December 31, 2079, and a precision of 1/300th of a second (3.33 milliseconds).
- datetime4 returns a 4-byte datetime datatype with a range of legal values from January 1, 1900, to June 6, 2079, and a precision of one minute.
- char_iso returns character data in the format *yyyy-mm-dd*.
- char_usa returns character data in the format *mm/dd/yyyy*.
- char_eur returns character data in the format *dd.mm.yyyy*.
- char_jis returns character data in the format *yyyy-mm-dd*.
- char_odbc returns character data in the format *yyyy-mm-dd.*

Comments

If the DATE value is outside the range of the Adaptive Server DATETIME, the DateTimeConvertError property is used to determine the desired datatype conversion.

DateTimeResults

Specifies the Open Client and Open Server datatype to which an access service converts ODBC TIMESTAMP results.

Syntax

```
DateTimeResults=[datetime | datetime4 | char_iso | char_usa | char_eur
| char_jis | char_odbc]
```

Default

datetime

Values

- datetime returns an 8-byte datetime datatype with a range of legal values from January 1, 1753, to December 31, 2079, and a precision of 1/300th of a second (3.33 milliseconds).
- datetime4 returns a 4-byte datetime datatype with a range of legal values from January 1, 1900, to June 6, 2079, and a precision of one minute.
- char_iso returns character data in the format *yyyy-mm-dd-hh.mm.ss.nnnnnn*.

- char_usa returns character data in the format *mm/dd/yyyy hh:mm* AM or PM.
- char_eur returns character data in the format *dd.mm.yyyy hh.mm.ss*.
- char_jis returns character data in the format *yyyy-mm-dd hh:mm:ss*.
- char_odbc returns character data in the format *yyyy-mm-dd hh:mm:ss.nnnnnn*.

Comments

- Use char_iso to retain the most precision.
- You can convert ODBC TIMESTAMP to one of the character formats to retain more precision. CS_DATETIME has less precision (1/300ths of a second) than ODBC TIMESTAMP, which can have a precision of up to six fractional places.
- A string representation of an ODBC TIMESTAMP starts with a digit and has a length of at least 16 characters. The complete string representation of an ODBC TIMESTAMP has the following form:
  yyyy-mm-dd-hh:mm:ss.nnnnnn

  Trailing blanks can be included.
- You can omit leading zeros from the month, day, and hour part of the ODBC TIMESTAMP. Also, you can truncate microseconds or omit them entirely. If you choose to omit any digit of the microseconds portion, an implicit specification of 0 is assumed.

DecimalResults

Specifies the Open Client and Open Server datatype to which an access service converts DECIMAL results.

Syntax

```
DecimalResults=[autoconvert | int | float | real | char | money | money4 |
bcd]
```

Default

autoconvert

Values

- autoconvert allows the access service to choose the appropriate datatype to return according to the following conversion scheme:
  - If scale = 0 and precision is less than or equal to 9, the access service returns CS_INT.
  - If scale is less than or equal to 4, and precision minus scale is less than or equal to 19, the access service returns CS_MONEY.
  - If scale is greater than 2, and precision minus scale is greater than 14, the access service returns CS_FLOAT.

- int returns a 4-byte integer type.
- float returns an 8-byte float type.
- real returns a 4-byte float type.
- char returns a character type. However, decimal points are not aligned consistently with those in the ODBC DECIMAL columns.
- money returns an 8-byte money type.
- money4 returns a 4-byte money type.
- bcd is valid only if you have columns described in binary coded decimal (BCD) format. The access service returns BCD columns as CS_BINARY or CS_VARBINARY with the following format:
    - If precision is even, the first nibble is 0.
    - Intervening digits are represented in BCD format with one nibble per digit.
    - The final nibble indicates the sign: C is positive and D is negative.
    - No indication of decimal position is given. The client application is responsible for determining decimal position.

Comment

In some ODBC data sources, the ODBC decimal type is not supported. In such cases, it is not supported as a Sybase datatype.

FloatResults

Specifies the Open Client and Open Server datatype to which an access service converts FLOAT results.

Syntax

```
FloatResults=[float | real | char]
```

Default

float

Values

- float returns an 8-byte float type.
- real returns a 4-byte float type.
- char returns a character type.

Int2Results

Specifies the Open Client and Open Server datatype to which an access service converts SMALLINT results.

Syntax

```
Int2Results=[smallint | char]
```

Default	smallint
Values	• smallint returns a 2-byte integer type. • char returns a character type.

Int4Results

Specifies the Open Client and Open Server datatype to which an access service converts INTEGER results.

Syntax	`Int4Results=[int | char]`
Default	int
Values	• int returns a 4-byte integer type. • char returns a character type.

RealResults

Specifies the Open Client and Open Server datatype to which an access service converts REAL results.

Syntax	`RealResults=[float | real | char]`
Default	real
Values	• float returns an 8-byte float type. • real returns a 4-byte float type. • char returns a character type.

Note DirectConnect ignores the char setting.

TimeResults

Specifies the Open Client and Open Server datatype to which an access service converts TIME results.

Syntax	`TimeResults=[ datetime | datetime4 | char_iso | char_usa | char_eur | char_jis | char_odbc ]`
Default	datetime

Values

- datetime returns an 8-byte datetime datatype with a range of legal values from January 1, 1753, to December 31, 9999, and a precision of 1/300th of a second (3.33 milliseconds).
- datetime4 returns a 4-byte datetime datatype with a range of legal values from January 1, 1900, to June 6, 2079, and a precision of one minute.
- char_iso returns character data in the format *hh.mm.ss.*
- char_usa returns character data in the format *hh:mm AM or PM.*
- char_eur returns character data in the format *hh.mm.ss*.
- char_jis returns character data in the format *hh:mm:ss.*
- char_odbc returns character data in the format *hh:mm:ss.nnnnnn.*

TinyIntResults

Specifies the Open Client and Open Server datatype to which an access service converts TinyIntResults.

Syntax

```
TinyIntResults=[ smallint, tinyint ]
```

Default

smallint

Values

- smallint returns an 2-byte integer.
- tinyint returns a 1-byte integer.

XNLChar

Specifies the maximum size of both char and binary results. If the maximum size is exceeded, the datatype is promoted to text and image, respectively.

Syntax

XNLChar=*integer*

Default

256

Values

integer is a valid number between 256 - 2147483647 (two gigabytes).

Comments

Sybase recommends that this value match the maximum size of the char and binary datatypes of the back-end database. It is common for this limit to be the same for the char and binary datatypes.

XNLVarChar

Specifies the maximum size of both varchar and varbinary results. If the maximum size is exceeded, the datatype is promoted to text and image, respectively.

Syntax	XNLVarChar=*integer*
Default	256
Values	*integer* is a valid number between 256 - 2147483647.
Comments	Sybase recommends that the value match the maximum size of the varchar and varbinary datatypes of the back-end database. It is common for this limit to be the same for the varchar and varbinary datatypes.

Logging properties

Logging properties control whether access service data is recorded in the DirectConnect server log file. The log statistics properties record similar types of data. You can use these properties independently or in combination to record statistics.

The following table describes the logging statistics properties.

Table 2-1: Logging statistics properties

Property Name	Description
LogConnectionStatistics	Records accumulated statistics about requests made by each client connection. Statistics are recorded when the client disconnects.
LogRequestStatistics	Records statistics about individual SQL requests.
LogServiceStatistics	Records accumulated statistics about requests made by all connections to this access service.
LogSvclibStatistics	Records accumulated statistics about requests made by connections to all access services within this access service library.
LogTransferStatistics	Records statistics about individual transfer requests.

The following table shows the statistics recorded by the LogConnectionStatistics, LogRequestStatistics, and LogSvclibStatistics properties.

Table 2-2: Logging statistics data

Log Field Data	Description
Buffer size	The number of bytes of SQL (after transformation) in the SQL request sent to the database
Service processing time	The elapsed time in seconds from when the access service receives a SQL statement until it sends the statement to the database
DBMS processing time	The elapsed time in seconds from when the access service sends the SQL statement to the database until the database returns the first result row to the client application
Time to receive rows	The elapsed time in seconds from when the database sends the first result row to the client application until the client application receives the last result row (in the format *ss.nnn*)
Total processing time	The elapsed time in seconds from when the access service receives the SQL statement until the client application receives the last result row
Number of rows returned	The number of result rows returned to the client application
Number of conversion errors	The number of result rows that contain data conversion errors
Number of kilobytes returned	The number of kilobytes returned to the client application (to a scale of 3 in the format *n.nnn*)
Number of events	The total number of events that occurred during the time period (LogSvclibStatistics and LogConnectionStatistics only)
Number of successful connections	The total number of successful client connections that occurred during the time period (LogSvclibStatistics only)
Maximum number of client connections	The greatest number of client connections at any given time during the time period (LogSvclibStatistics only)
Command type	The first command (token) from the SQL buffer (LogRequestStatistics only)

For information about the server log file, see the ECDA and Mainframe Connect *Server Administration Guide* for DirectConnect.

The subsection heading and a list of the properties must appear in the access service library configuration file as shown:

```
{Logging}
LogConnectionStatistics=
LogReceivedSQL=
LogRequestStatistics=
LogServiceStatistics=
LogSvclibStatistics=
LogTargetActivity=
LogTransferStatistics=
LogTransformedSQL=
```

LogConnectionStatistics

Specifies whether the access service records accumulated statistics about each connection.

Syntax	LogConnectionStatistics=[no | yes]
Default	no
Values	• no means connection statistics are not recorded. • yes means connection statistics are recorded.
Comments	• Connection statistics are recorded in the server log file when the client disconnects from the access service. • You can use this property to monitor the activity of particular clients. • For a list of recorded statistics data, see Table 2-2 on page 41.

LogReceivedSQL

Specifies whether the access service records SQL statements as the statements are received from client applications.

Syntax	LogReceivedSQL=[no | yes]
Default	no
Values	• no means the access service does not record SQL statements as the statements are received. • yes means the access service records the SQL statements as received.

LogRequestStatistics

Specifies whether the access service records statistics about each SQL request.

Syntax	LogRequestStatistics=[no | yes]
Default	no
Values	• no means the access service does not record statistics about each request.

	• yes means the access service records statistics about each request.
Comments	• Use this property to: • Aid performance tuning on a specific type of request • Analyze data throughput • Monitor the types of requests by users • For a list of recorded statistics data, see Table 2-2 on page 41.

LogServiceStatistics

Specifies how often the access service records accumulated statistics about the access service.

Syntax	LogServiceStatistics=*integer*
Range	0–2147483646
Default	0
Values	• *integer* is a number of seconds. • A value of 0 specifies that the access service does not record access service statistics.
Comment	Use this property to: • Monitor the load on a particular access service • Monitor usage of a particular access service

LogSvclibStatistics

Specifies how often the access service library records accumulated statistics about connection requests to all access services associated with this access service library during the reporting interval. This is an access service library property that applies to the access service library as a whole.

Syntax	LogSvclibStatistics=*integer*
Range	0–2147483646
Default	0
Values	• *integer* is a number of seconds. • A value of 0 specifies that the access service library does not record statistics in the server log file.

Comments

- Use this property to:
  - Monitor load on the entire access service library
  - Monitor load on the target database through the DirectConnect server
- If you enable both LogSvclibStatistics (service library level) and LogServiceStatistics (service level) properties, Sybase recommends that you set the LogSvclibStatistics property to the same property value as the LogServiceStatistics or a multiple thereof. If you use DirectConnect Manager to change these two property values, set the LogSvclibStatistics property last for better synchronization.
- If the LogSvclibStatistics property value is greater than 0, DirectConnect records totals of the statistics for all access services in the access service library.
- For a list of recorded statistics data, see Table 2-2 on page 41.

LogTargetActivity

Specifies whether the access service records interactions between the access service and the target database. A file is created for each connection.

Syntax

```
LogTargetActivity=[no | yes]
```

Default

no

Values

- no means the access service does not record access service interactions with the target database.
- yes means the access service records the following access service interactions with the target database:
  - Login
  - Logout
  - Requests sent
  - Results received

LogTransferStatistics

Specifies whether the access service records statistics about transfers.

Syntax

```
LogTransferStatistics=[no | yes]
```

Default

no

Values

- no means the access service does not record transfer statistics.
- yes means the access service records transfer statistics.

Comments

The access service records the transfer statistics shown in the following table.

Table 2-3: LogTransferStatistics data

Log Field Data	Description
Buffer size	The number of bytes in the transfer statement received by the access service.
Transfer setup time	The elapsed time in seconds from when the access service receives a transfer statement until it issues the source select against the secondary database. This includes connection time to the secondary database and the time the access service takes to obtain datatype information for target columns.
Source DBMS processing time	The elapsed time in seconds from when the access service issues the select part of the transfer statement against the source database until it receives the first result row from the source of the transfer.
Time to transfer rows	The elapsed time in seconds from when the source database returns the first row to the access service until the last row is inserted into the target of the transfer.
Total processing time	The elapsed time in seconds from when the access service receives the transfer statement until the last row of the transfer data is inserted into the target database.
Number of rows transferred	The number of rows transferred.
Number of conversion errors	The number of rows that contain data conversion errors.
Number of kilobytes returned	The total number of kilobytes transferred (to a scale of 3 in the format *n.nnn*).
Command type	The first token (command) from the SQL buffer. This is always a transfer.

LogTransformedSQL

Specifies whether the access service records SQL as it is transformed and sent to the target database.

Syntax

```
LogTransformedSQL= [no | yes]
```

Default

no

Values

- no means the access service does not record SQL as it is transformed and sent to the target database.

- yes means the access service records SQL as it is transformed and sent to the target database.

Target interaction properties

These properties control how an access service interacts with the target database.

The subsection heading and a list of the properties must appear in the access service library configuration file as shown:

```
{Target Interaction}
Allocate=
DelimitSqlRequests=
IsolationLevel=
QuotedStringDelimeter=
ReturnNativeError=
SQLOdbcCursors=
SQLTransformation=
StopCondition=
TargetDBMS=
TargetDecimalSeparator=
```

Allocate

Controls when an access service allocates conversations with the target database system.

Syntax	Allocate=[connect | request]
Default	connect
Values	• connect specifies an access service to allocate the conversation when the client connects, and holds it open for the duration of the client connection.

- request specifies an access service to allocate a new conversation each time the client application sends a request, and deallocates the conversation after each request.

Note There is a large performance penalty when using the request setting.

DelimitSqlRequests

Allows statements to be batched and executed as a single request to the target DBMS. Statements are delimited with a semicolon (;).

Syntax

```
DelimitSqlRequests=[yes | no]
```

Default

yes

Values

- no specifies statements that will be batched and executed as a single request to the target DBMS, allowing the creation of stored procedures and triggers.
- yes specifies statements that contain semicolons will be broken down with each delimited statement being sent separately to the target DBMS.

IsolationLevel

Controls the level of locking and record access to ODBC-accessible tables.

Syntax

```
IsolationLevel=[ur | cr | rr | sr | vr | no]
```

Default

no

Values

Following are descriptions of isolation levels from the *Microsoft ODBC 3.5 Programmer's Reference and SDK Guide*:

- Dirty read: Transaction 1 changes a row. Transaction 2 reads the changed row before transaction 1 commits the change. If transaction 1 rolls back the change, transaction 2 will have read a row that is considered to have never existed.
- Nonrepeatable Read: Transaction 1 reads a row. Transaction 2 updates or deletes that row and commits the change. If transaction 1 attempts to reread the row, it receives different row values or discovers that the row has been deleted.

- Phantom: Transaction 1 reads a set of rows that satisfy some search criteria. Transaction 2 inserts a row that matches the search criteria. If transaction 1 re-executes the statement to read the rows, it receives a different set of rows.

Acceptable values are as follows:

- ur (uncommitted read): Dirty reads, nonrepeatable reads, and phantoms are possible.
- cr (committed read): Dirty reads are not possible. Nonrepeatable reads and phantoms are possible.
- rr (repeatable read): Dirty reads and nonrepeatable reads are not possible. Phantoms are possible.
- sr (serializable): Transactions can be serialized. Dirty reads, nonrepeatable reads, and phantoms are not possible. This is usually implemented by using locking protocols that reduce concurrency.
- vr (versioning): Transactions can be serialized, but this value provides higher concurrency. Dirty reads are not possible. This is usually implemented by using nonlocking protocols, such as record versioning.
- no (none): Use the ODBC driver default level.

QuotedStringDelimiter

Specifies the character used for quoted strings.

Syntax	QuotedStringDelimiter=*char*
Range	0–1 characters
Default	(single quote)
Values	*char* is the quoted string delimiter for your locale.

ReturnNativeError

Allows a non-localized native error message and a native error severity to be returned to the client.

Syntax	ReturnNativeError = [yes | no]
Default	no
Values	• yes specifies non-localized native error messages are returned to the client.

- no specifies non-localized native error messages are not returned to the client.

SQLOdbcCursors

The SQLOdbcCursors target interaction configuration property is added to control ODBC cursor library usage.

Syntax

```
SQLOdbcCursors=[if_needed | odbc | driver | default]
```

Default

if_needed

Values

- if_needed specifies use of the cursor library only if it is needed.
- odbc specifies use of the ODBC cursor library.
- driver specifies use of the cursor management capability of the driver.
- default specifies to use the cursor library only if needed.

SQLTransformation

Specifies the mode the access service uses for SQL transformation.

Syntax

```
SQLTransformation=[passthrough | sybase | tsql0 | tsql1 | tsql2]
```

Default

passthrough

Values

- passthrough specifies an access service to send all SQL statements to the database system as received, without transformation. A client application uses passthrough mode to gain direct access to DBMS capabilities.
- sybase specifies an access service to perform SQL transformation of selected statements. It also allows the use of multi-part table names with the view command in SQL statements.
- For backward compatibility *only*, the access service also accepts the following parameters:

  ```
  [tsql0 | tsql1 | tsql2]
  ```

  - Setting to tsql0 mode is the same as setting to passthrough mode.
  - Setting to tsql1 mode, the database gateway expects SQL written for DB2 but performs minor Adaptive Server adaptations.
  - Setting to tsql2 modes, implies Transact-SQL, providing less than the desired results in every situation. It performs maximal request transformation, including parsing at the database gateway.

For information about transformation modes, see Chapter 6, “Issuing SQL Statements.”

Comments: For additional information, refer to the *MDI Database Gateway User’s Guide* for your target database and platform.

StopCondition

Specifies under what conditions an access service stops processing results.

Syntax:

```
StopCondition=[error | none | warning]
```

Default: error

Values:

- error specifies an access service to stop processing results *only* when an error occurs.
- none specifies an access service not to stop processing results when errors or warnings occur.
- warning specifies an access service to stop processing results when an error or warning occurs.

TargetDBMS

Specifies whether SQL transformation will be done to specific dialects of DB2 for z/OS, or DB2 for Windows and UNIX.

Syntax:

```
TargetDBMS=[notused | UDBLAN | UDBOS390 | UDBAS400]
```

Default: notused

Values:

- notused specifies no SQL transformation to specific dialects.
- UDBLAN specifies SQL transformation to specific dialects for DB2 for Windows and UNIX.
- UDBOS390 specifies SQL transformation to specific dialects for DB2 for z/OS.
- UDBAS400 specifies SQL transformation to specific dialects for DB2 for AS/400.

Comments:

- Affects the generation of DB2 DDL statements, such as create table and alter table.

- Setting the value to UDBLAN or UDBOS390 indicates which create table syntax to use when in sybase translation mode. The database part of the three-part table name is appended with or without the IN DATABASE clause. UDBLAN causes the translation to drop the DATABASE symbol from the IN DATABASE clause.

TargetDecimalSeparator

Specifies the character delimiter for the fractional part of decimal numbers passed between the access service and the target DBMS.

Syntax	TargetDecimalSeparator=*char*
Range	0–1 characters
Default	. (period)
Values	• *char* indicates the character used by the target as the decimal delimiter. • A period indicates that the target database uses a period as the decimal delimiter.

Tracing properties

Tracing properties control whether access service data is written to the DirectConnect server trace file. For information about the DirectConnect server trace file, see the ECDA and Mainframe Connect *Server Administration Guide* for DirectConnect.

Because tracing degrades performance, use tracing only when instructed to do so by Sybase Technical Support.

Warning! To provide Sybase Technical Support with all the necessary trace data, the server trace file is allocated a maximum of 20MB of space. When the server trace file exceeds the maximum, it is copied to a file with the same file name and with an "*_old*" extension (<*filename*>*_old*). See the ECDA and Mainframe Connect *Server Administration Guide* for DirectConnect for suggestions for deleting or backing up old log and trace files.

The subsection heading and a list of the properties must appear in the access service library configuration file as shown:

```
{Tracing}
TraceEvents=
TraceInterface=
TraceTarget=
```

TraceEvents

Specifies whether the access service traces the event handler layer of the access service library.

Syntax TraceEvents=[no | yes]

Default no

Values
- no means the access service does not trace the event handler layer.
- yes means the access service traces the event handler layer.

TraceInterface

Specifies whether the access service traces the interface layer of the connectivity library.

Syntax TraceInterface=[no | yes]

Default no

Values
- no means the access service does not trace the interface layer.
- yes means the access service traces the interface layer.

Comments
- The *TraceFilename* file contains the interface layer tracing.

TraceTarget

Specifies whether the access service traces the implementation layer of the access service library.

Syntax TraceTarget=[no | yes]

Default no

Values
- no means the access service does not trace to the target layer.
- yes means the access service traces to the target layer.

Transfer properties

These properties control how an access service performs transfer processing.

The subsection heading and a list of the properties must appear in the access service library configuration file as shown:

```
{Transfer}
BulkCommitCount=
TransferBatch=
TransferErrorAction=
TransferErrorCount=
TransferExpress=
```

BulkCommitCount

Specifies how many bulk transfer rows are transferred before a commit is sent to the target connection.

Syntax	BulkCommitCount=*integer*
Range	0–32767
Default	0
Value	*integer* is a number of rows transferred before a commit is issued (between commits).

TransferBatch

Specifies how many destination-template transfer clauses can be batched in one request.

Syntax	TransferBatch=*integer*
Range	0–32767
Default	1
Value	*integer* is a number of destination-template clauses.

Comments

- The access service builds the designated number of destination templates in its request buffer before executing each request.
- If the value is 0, the access service sends all statements that fit the request buffer.
- An invalid value defaults to 0.
- For more information about destination-template transfer, see Chapter 10, "Using Destination-Template Transfer."

TransferErrorAction

Specifies whether the transfer batch is to be rolled back or committed when an error occurs.

Syntax

```
TransferErrorAction=[noaction | rollback]
```

Default

noaction

Values

- Noaction specifies the transfer batch to be committed when an error occurs.
- Rollback issues a rollback when an error occurs, the TransactionMode is long, and the TransferErrorCount is exceeded.

Comments

To use the rollback property, you must set the following properties:

- TransferErrorAction property to rollback
- TransferErrorCount property to a value greater than zero
- TransactionMode property to long

TransferErrorCount

Specifies how many error rows are allowed during destination-template transfer before processing stops.

Syntax

TransferErrorCount=*integer*

Range

0–32767

Default

0

Value

integer is a number of error rows.

Comment

A value of 0 means transfers do not stop processing, regardless of the number of errors encountered.

TransferExpress

Specifies whether the bulk transfer statements are handled by express transfer processing.

Syntax

```
TransferExpress=[no | yes]
```

Default

no

Values

- no means that standard bulk processing is to be done.
- yes means that express transfer processing is to be done.

Comments	Express transfer requires that the transfer statement secondaryname must match a DSN in the *ODBC system information* file.

CHAPTER 3

Configuring Access Services to Work with Related Products

This chapter provides instructions for setting up Adaptive Server Enterprise (ASE), the Component Integration Services functionality in ASE (ASE/CIS), and Replication Server to use with a DirectConnect access service.

This chapter contains the following topics:

Setting up Adaptive Server Enterprise

To set up ASE for remote procedure calls (RPCs) against a DirectConnect access service, you must first create an ASE user ID and password that match a DirectConnect access service user ID and password. For instructions on creating the ID and password, see your ASE documentation.

❖ **To set up ASE**

1. Configure Adaptive Server for remote access.
2. Define each DirectConnect access service as a remote server.
3. Define connectivity between Adaptive Server and the DirectConnect access service.

Note To set up connectivity for a remote server, see the appropriate Adaptive Server Enterprise documentation for the platform on which Adaptive Server Enterprise is installed.

1. Configure for remote access

To configure Adaptive Server for remote access, log in to Adaptive Server as sa and check the current sp_configure setting using the following command:

```
sp_configure 'remote access'
```

If the returned value is 1, Adaptive Server is configured for remote access.

❖ **If the returned value is *0***

1 Enter:

   ```
   sp_configure 'remote access',1
   ```

2 Restart Adaptive Server.

2. Define the access service as a remote server

❖ **Enter each DirectConnect access service name in the Adaptive Server Enterprise *SYSSERVERS* table**

1 Enter each DirectConnect access service name in the Adaptive Server Enterprise SYSSERVERS table

   ```
   sp_addserver service_name service_class
   ```

   where:

   - *service_name* is the name of the DirectConnect access service you want to set up as a remote server.
   - *service_class* must be direct_connect.

   The name is case sensitive and must match the name you used to define connectivity between Adaptive Server Enterprise and the DirectConnect access service.

2 Verify that the DirectConnect access service is successfully defined as a remote server:

   ```
   sp_helpserver server_name
   ```

   This command returns a result set with a list of the services defined to Adaptive Server.

3 Use the sp_addexternlogin system procedure to add an external login password to allow access to the DirectConnect access services.

4 Verify ASE/CIS connectivity using the ASE/CIS extension to Transaction-SQL connect to *server_name*. This is the same server name you created in a previous step using the sp_addserver system procedure. This command establishes a passthrough mode connection to the DirectConnect access service. The passthrough mode remains in effect until you issue a disconnect command.

5 Enable location transparency of remote data through remote object mapping:

- Use sp_addobjectdef to define the storage locations of remote objects.
- Use either create table or create existing table (as applicable) to map remote table schema to ASE/CIS.

6 Use select to perform joins across the DirectConnect access service after following all the previous steps.

For more information about the process plus instructions for configuring and tuning ASE/CIS, see the *Component Integration Services Guide* for Adaptive Server Enterprise.

RPC issues

When Adaptive Server issues an RPC to a DirectConnect access service, it attempts to connect using a service name identical to the Adaptive Server Enterprise identifier in the *interfaces* file (for UNIX) or the *sql.ini* file (for Windows).

To support Adaptive Server RPC events, perform either one of the following procedures:

- Define a service within the access service library using the same name as Adaptive Server.
- Set up service name redirection to redirect the Adaptive Server to the appropriate service.

3. Define connectivity between ASE and DirectConnect access service

In the *sql.ini* file (for Windows) or the *interfaces* file (for UNIX), add a QUERY section with the same name and *ConnectionSpec* that was added in step 2 (*service_name*).

For more information, see the *System Administration Guide* for Sybase Adaptive Server and the Adaptive Server Enterprise *Reference Guide*.

Setting up DirectConnect for ASE/CIS

You can configure DirectConnect to be configured by ASE/CIS to transparently access data on a remote server, such as a DirectConnect access service, by performing the following.

Note Be sure you are running the latest ASE/CIS update (ESD) for your platform.

❖ **To set up DirectConnect for ASE/CIS**

1. Configure the following properties for DirectConnect for Microsoft SQL Server:
   - QuotedIdentifier=*on*
   - SqlTransformation=*sybase*
   - CSPColumnODBCVersion=*2* (ASE/CIS Version 12.0)
   - Use default, CSPColumnODBCVersion=*3* (ASE/CIS 12.5 and up)
2. Configure the Microsoft SQL Server driver:
   - For Windows:
     Choose "enabled quoted identifiers" for Microsoft SQL Server driver in the ODBC administrator
   - For UNIX:
     Add the configuration property QuotedId=yes in the ODBC system information file (*odbc.ini* for UNI

Using ASE/CIS with the ASE transaction model

ASE/CIS requires a DirectConnect access service to support particular aspects of the Adaptive Server transaction model. If the DirectConnect access service is configured correctly for ASE/CIS, and if it recognizes the ASE/CIS client application, it supports the transaction model. The model includes the following statements:

- begin tran
- prepare tran

- commit or rollback

Note This is a limited implementation of the Adaptive Server transaction model and is available only for ASE/CIS. Other applications cannot use these features.

To process ASE/CIS transactions, you must set the transaction mode and transform level parameters. For instructions, see the Adaptive Server Enterprise *Component Integration Services Guide*.

If all conditions are met, the transaction statements are handled as follows:

begin tran statement

The DirectConnect access service does the following:

- Sets a flag so that it knows it is in an ASE/CIS transaction
- Sets the transaction mode to long
- Returns a successful status, as if a set command were executed

Other considerations are as follows:

- Nested transactions are not required, nor are they supported.
- ASE/CIS does not issue this statement if outstanding changes were not committed or rolled back.
- The DirectConnect access service issues a commit prior to handling the statement.

prepare tran statement

If the DirectConnect access service is in an ASE/CIS transaction, it does the following:

- Ignores this statement
- Returns a successful status, just as if a set command were executed

This statement is issued in preparation for a commit statement.

commit or rollback statement

If the DirectConnect access service is in an ASE/CIS transaction, it performs the following:

- Resets the flag that indicates it is in an ASE/CIS transaction
- Resets the mode to short
- Returns a success or failure status of the commit or rollback statement to ASE/CIS

ASE/CIS expects the DirectConnect access service to be in short transaction mode unless:

- ASE/CIS opened one or more read-only cursors.
- ASE/CIS issued a begin tran but did not issue a commit or rollback.

Because the DirectConnect access service automatically switches to long transaction mode when a cursor is declared or when a dynamic event prepare occurs, it backs out of this mode when the last cursor and prepared statement are deallocated and not in a begin tran block.

Note ASE/CIS always issues dynamic events within its own transaction, so it is not necessary to back out when the last dynamic statement is freed.

In the event of a rollback, the DirectConnect access service issues a commit and sets the transaction mode to short.

Setting up DirectConnect for use with Replication Server

Replication Server provides several scripts that can help you set up a connection from the target database to Replication Server using a DirectConnect access service that is configured with the following settings:

- TransactionMode = *long*
- ReturnNativeError = *yes*
- SQLTransformation = *sybase*
- Allocate = *connect*

For more information, see the Replication Server *Administration Guide*, specifically Chapter 6, “Managing Database Connections,” and Chapter 12, “Customizing Database Operations.”

CHAPTER 4

Querying and Setting Operating Values

This chapter describes how to use global variables and set statements to query and set operating values for the property and processing values for your client connection.

This chapter contains the following topics:

Querying global variables

A user or client application can query a global variable to find the property and processing values that affect that client connection.

A global variable represents one of the following:

- The current value of a set statement
- Information about the processing state of a connection

Global variables are preceded by two "@@" characters and are not case sensitive. To query a global variable, issue a SQL statement in the form:

```
    select @@variable_name
```

where *variable_name* is the name of the relevant global variable.

Other rules that apply are as follows:

- All global variables can be queried regardless of the SQL transformation mode in effect for the connection.
- The select statement must be the only one in a batch.
- The statement can be terminated with a semicolon.

Example

The following is an SQL statement to query a global variable for the Client Interaction property, SendWarningMessages:

```
select @@SendWarningMessages
```

Issuing set statements

A user or client application can issue a set statement to change values that affect only the current client connection. These values remain in effect only for the duration of the client connection or until another set statement is issued.

Access service set statements are not case sensitive. To set an access service configuration property value or processing value for the current connection, issue a set statement as follows:

```
set { property_name | processing_name } value
```

where:

- *property_name* is the name of the configuration property.
- *processing_name* is the name of the processing option.
- *value* is a valid value for the configuration property.

Example

The following is a set statement for the Client Interaction property, SendWarningMessages:

```
set SendWarningMessages {no|yes}
```

CHAPTER 5 Managing Transactions

This chapter describes the transaction management processing flow and provides tables showing the effects of certain configuration properties on transaction processing.

This chapter contains the following topics:

Transaction processing terms

To understand the transaction processing flow, you should first know the following terms:

- Request
- Unit of work
- Transactions

Each is explained briefly in the following subsections.

Request

A request is equivalent to the contents of the buffer. It can contain a single statement or a batch of statements. During a request, the client application gives up control to the Database Management System (DBMS) and waits for a response.

Unit of work

A unit of work is one or more requests that are committed or rolled back as a group. If all the requests process successfully, the unit of work is committed, and the requested changes to the database are permanent.

Transactions

A transaction is a set of SQL statements terminated with a commit or rollback operation. It is equal to a unit of work and can span many requests.

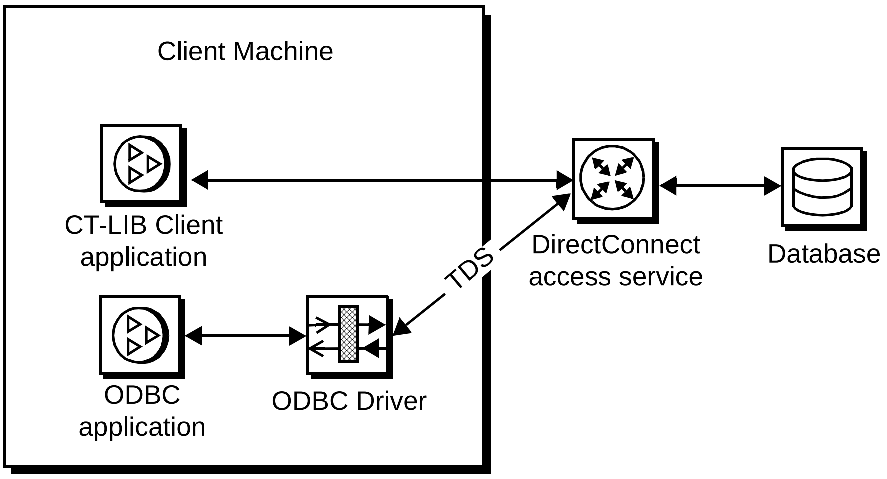
Request processing flow

Request processing consists of the following steps:

1. The client application issues a request (for example, a select statement).
2. The client Application Program Interface (API) receives the request and sends it to the access service.
3. The DirectConnect access service receives the request, transforms it as specified in its configuration, and executes it.
4. After the request processes, the access service converts target datatypes to Open Server datatypes and returns the results to the client application.
5. The client application disconnects from the access service.

The following figure shows the processing flow from the client application through the DirectConnect access service to the database.

Figure 5-1: Request processing flow

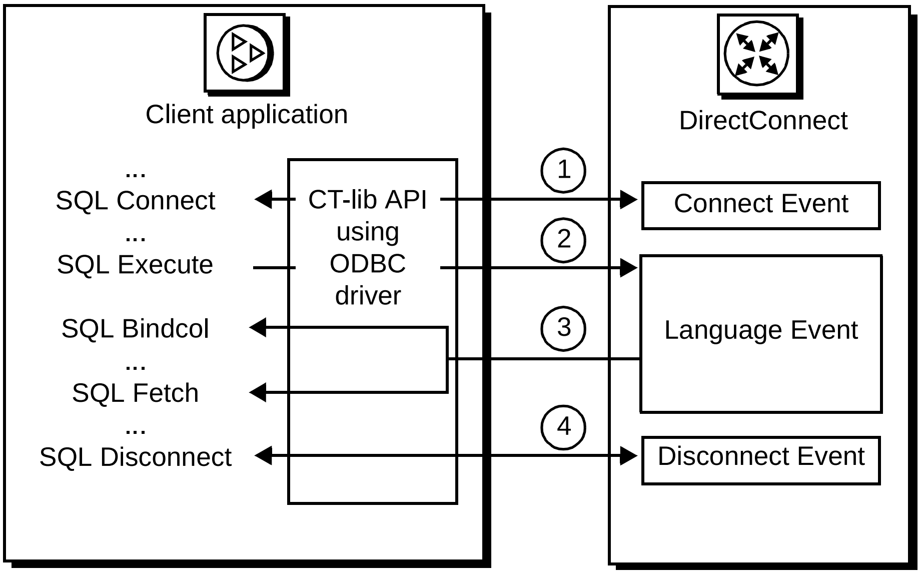


For more information about Open Client and Open Server products, see the Open Client *Client-Library/C Reference Manual* and the Open Server *Server-Library/C Reference Manual*.

ODBC client API processing

Sybase provides a DirectConnect ODBC driver that allows ODBC applications to access data through DirectConnect. The following figure shows the interaction between an ODBC client application and DirectConnect.

Figure 5-2: ODBC client API example



The ODBC API processing flow includes the following steps:

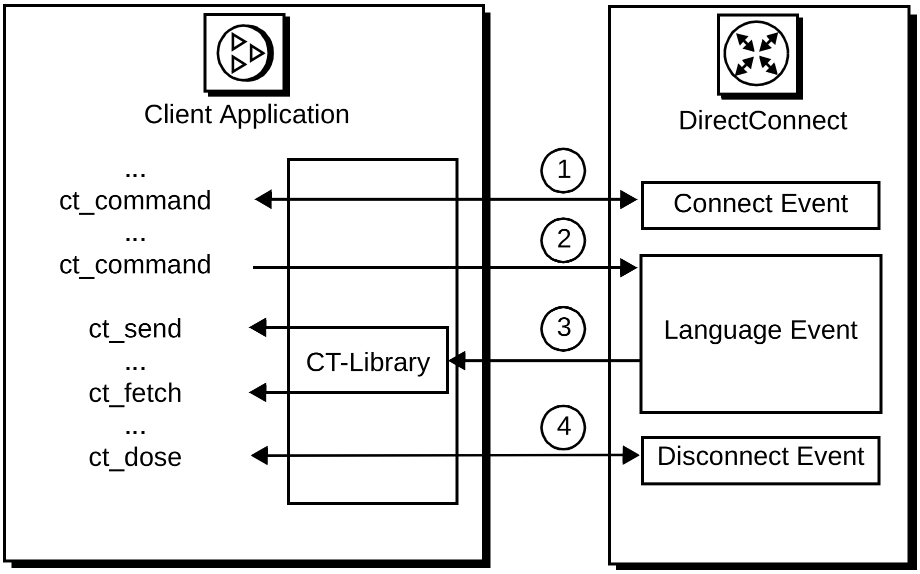
1. The ODBC client application initiates a connect event to a specific DirectConnect access service using the SQLConnect call. The ODBC driver uses the CT-Library API.
2. The ODBC application builds and executes the request using the SQLExecDirect function. This initiates a language event in the DirectConnect server.
3. The ODBC API uses SQLBindCol to assign local variables to specific columns. The SQLFetch call returns the resulting data to the application.
4. The ODBC API terminates the request with a SQLDisconnect call. This initiates a disconnect event in the DirectConnect server.

Procedure calls for the ODBC API are described in the Microsoft *ODBC 3.5 Programmer's Reference and SDK Guide*.

CT-Library client API processing

The following figure shows the interaction between a Sybase Open Client CT-Library API and DirectConnect.

Figure 5-3: CT-Library client API example



The CT-Library client API processing flow includes the following steps:

1. When a CT-Library application issues a request, it uses the ct_connect call to initiate a connect event to a specific DirectConnect access service.
2. The CT-Library API executes the request using the ct_command and ct_send calls. This initiates a language event in the DirectConnect server.
3. The CT-Library API uses ct_fetch to return the requested results to the client application.
4. The CT-Library API terminates the request with ct_close. This call initiates a disconnect event in the DirectConnect server.

Procedure calls for the CT-Library API are described in the Sybase Open Client *Client-Library/C Reference Manual*.

Note DirectConnect does not support the use of text or image pointers. Consequently, ct_data, ct_get_data, and ct_put_data, when used with text or image pointers, are not supported.

Managing processing results

You can manage processing by using the Allocate, StopCondition, and TransactionMode configuration properties together.

Allocate configuration property

The Allocate configuration property determines when a connection should be deallocated from the host server as follows:

- If Allocate is set to connect, the connection is kept until the client issues some type of deallocation.
- If Allocate is set to request, the connection is established at the start of a request, then dropped.

Allocate does not affect the transaction mode.

If Allocate is set to request and the transaction mode is long, the connection stays available until a batch ends on a commit or rollback statement or exit. If the batch ends on a commit or rollback statement, the transaction ends properly. If the batch ends with exit, the connection also ends and any uncommitted SQL statements are rolled back.

If Allocate is set to request and the transaction mode is short, after the request is processed, all SQL statements in the request are committed and the connection is dropped. If the request ends with a rollback statement, the SQL statements in the request are rolled back and the connection is dropped.

If Allocate is set to request, the transaction mode is short, and a begin transaction is encountered in the request, the SQL statements to that point are committed. The connection stays open, and the transaction mode is set to temporary long. Once a commit or rollback statement is encountered, the transaction mode reverts to short, and the allocation reverts to request.

While in temporary long transaction mode, the connection stays open until the transaction ends, even if the transaction spans multiple requests. The allocate on request is ignored until the transaction ends.

StopCondition configuration property

The StopCondition configuration property specifies whether the access service stops processing a request when it encounters an error or a warning. Valid values are as follows:

- error
- warning
- none

If you specify none, processing continues even when errors occur. This property is useful if you batch multiple statements in a request.

TransactionMode configuration property

The TransactionMode configuration property governs transaction behavior. You can set it to short or long.

Short transactions

If the transaction mode is set to short, the access service is responsible for controlling commitment of requests. After sending the request to the database, the access service automatically issues one of the following:

- A commit, if the request succeeds
- A rollback, if the request fails

The begin transaction phrase affects the behavior of short transactions as follows:

- If the begin transaction is issued prior to the end of the request, it triggers a commit of all previous statements in the request and temporarily sets the transaction mode to long.
- If the begin transaction is not issued prior to the end of a request, the request marks the end of the transaction, and the transaction is committed.
- You can issue a commit or rollback to end the temporary long mode. Doing this causes the transaction mode to revert to its status before the begin transaction was issued.
- If the batch is in temporary long mode and ends (exits) without a commit or rollback, the SQL statements are rolled back, and the transaction mode reverts to its original status.

Long transactions

If the transaction mode is set to long, you control when the transaction ends by issuing a commit or rollback. If the access service encounters a begin transaction, an error message is issued.

The client application issues a commit or rollback statement for each transaction. When the client application closes the connection, the access service issues a rollback before exiting.

Recommended actions for transaction management

The following table shows recommended actions with the different combinations of the Allocate, StopCondition, and TransactionMode configuration properties.

Table 5-1: State, event, action table for transaction management

Configuration property states and conditions	Event	Action

TransactionMode = *short*
Allocate = *request*
StopCondition = *error* or *warning*
	If StopCondition is encountered	Roll back transaction at StopCondition. Do not continue in request. End the connection.

	If end of request with no StopCondition	Commit transaction and end the connection.
	If begin transaction is encountered in request	Commit transaction. Switch to temporary *long* transaction mode and turn on begin transaction.
	If commit or rollback is encountered	Issue the commit or rollback.
	If cursor declare or dynamic prepare is encountered	Commit transaction. Switch to *long* transaction mode.

Configuration property states and conditions	Event	Action

TransactionMode = *short*
Allocate = *request*
StopCondition = *none*
	If error or warning is encountered	Do not roll back. Continue in request. Keep the connection until end of request.

	If end of request	Commit transaction and end the connection.
	If begin transaction is encountered in request	Commit transaction. Switch to temporary *long* transaction mode and turn on begin transaction.
	If commit or rollback is encountered	Issue the commit or rollback.
	If cursor declare or dynamic prepare is encountered	Commit transaction. Switch to *long* transaction mode.

TransactionMode = *short*
Allocate = *connect*
StopCondition = *error* or *warning*
	If StopCondition is encountered	Roll back transaction at StopCondition. Do not continue in request. Keep the connection.

	If end of request with no StopCondition	Commit transaction and keep the connection.
	If begin transaction is encountered in request	Commit transaction. Switch to temporary *long* transaction mode and turn on begin transaction.
	If commit or rollback is encountered	Issue the commit or rollback.
	If cursor declare or dynamic prepare is encountered	Commit transaction. Switch to *long* transaction mode.

TransactionMode = *short*
Allocate = *connect*
StopCondition = *none*
	If error or warning is encountered	Do not roll back. Continue in request and keep the connection.

	If end of request with no StopCondition	Commit transaction and keep the connection.
	If begin transaction is encountered in request	Commit transaction. Switch to temporary *long* transaction mode and turn on begin transaction.
	If commit or rollback is encountered	Issue the commit or rollback.
	If cursor declare or dynamic prepare is encountered	Commit transaction. Switch to *long* transaction mode.

Configuration property states and conditions	**Event**	**Action**

TransactionMode = *long* or temporary *long*
Allocate = *request* or *connect*
StopCondition = *error* or *warning*
	If StopCondition is encountered	Do not roll back transaction at StopCondition. Do not continue in request. Keep the connection.

	If end of request with no StopCondition	Keep the connection.
	If end of request with commit or rollback	End the connection.
	If begin transaction is encountered in request	If not in begin transaction block, set begin transaction. If in begin transaction block, then ignore. Issue message.
	If commit or rollback is encountered	Issue the commit or rollback. If in begin transaction block, end begin transaction.
	If cursor declare or dynamic prepare is encountered	Execute statement.
Same, plus cursor and dynamic statements all freed and in temporary *long* mode	If commit or rollback is encountered	Issue the commit or rollback. End begin transaction. Revert to previous transaction mode.
Same, plus cursor and dynamic statements active and in temporary *long* mode	If Free of a cursor or dynamic means that all cursors and dynamics are now freed	Execute free. If not in begin transaction block, revert to previous transaction mode.

TransactionMode = *long* or temporary *long*
Allocate = *request* or *connect*
StopCondition = *none*
	If StopCondition is encountered	Do not roll back transaction at StopCondition. Continue in request and keep the connection.

	If end of request with no StopCondition	Keep the connection.
	If end of request with commit or rollback	End the connection.
	If begin transaction is encountered in request	If not in begin transaction block, set begin transaction. If in begin transaction block, then ignore. Issue message.

TransactionMode = *long* or temporary *long*
Allocate = *request* or *connect*
StopCondition = *none*
	If commit or rollback is encountered	Issue the commit or rollback. If in begin transaction block, end begin transaction.

	If cursor declare or dynamic prepare is encountered	Execute statement.

Configuration property states and conditions	Event	Action
Same, plus cursor and dynamic statements all freed and in temporary *long* mode	If commit or rollback is encountered	Issue the commit or rollback. End begin transaction. Revert to previous transaction mode.
Same, plus cursor and dynamic statements active and in temporary *long* mode	If Free of a cursor or dynamic, all cursors and dynamics are now freed	Execute free. If not in begin transaction block, revert to previous transaction mode.

Troubleshooting

You can troubleshoot processing problems by using the DirectConnect server log and trace files.

Configuration properties control whether data is recorded in the server log and trace files for each logging and tracing option. To configure logging and tracing properties, edit the *dcany.cfg* configuration file or use DirectConnect Manager.

Tracing options

Tracing properties allow you to record troubleshooting information for Sybase Technical Support.

Warning! To provide Sybase Technical Support with all the necessary trace data, the server trace file will be allocated a maximum of 20MB of space. When the server trace file exceeds the maximum, it will be copied to a file with the same filename and with an "*_old*" extension (*<filename>_old*). See the ECDA and Mainframe Connect *Server Administration Guide* for DirectConnect for suggestions for deleting or backing up old log and trace files.

- For information about access service library and access service tracing property syntax, see "Tracing properties" on page 51.
- For information about DirectConnect server tracing properties and the server trace file, see the ECDA and Mainframe Connect *Server Administration Guide* for DirectConnect.

CHAPTER 6

Issuing SQL Statements

This chapter describes the SQL transformation modes that the DirectConnect access service uses to interpret data.

This chapter contains the following topics:

Overview

SQL transformation modes affect the way the DirectConnect access service modifies SQL statements that are written for one database but are processed against another.

The transformation mode primarily affects the way in which the access service treats incoming SQL statements, but it also affects these functional areas:

- Transaction
- Datatype handling
- set statements
- RSPs
- ASE/CIS interoperability
- ODBC driver interoperability

When you configure the access service with a specific transformation mode, that mode is effective for all client connections unless you use a set statement to alter it for a specific connection.

To make dialects appear as common SQL, the access service supports the following standard transformation modes:

- passthrough
- sybase

passthrough mode

Use passthrough mode when you want the client to have direct access to the capabilities of a DBMS target. The SQL dialect is for ODBC, not the actual DBMS dialect to which the ODBC driver is connected.

The access service performs virtually no SQL transformation. The passthrough mode converts carriage returns and line feeds. All other commands are passed directly to the ODBC driver, and the results are returned to the client.

Note In DirectConnect, you can prepare multiple SQL statements in a single statement. When you do this, be sure to separate the statements with semicolons.

sybase mode

Use sybase mode for maximum compatibility between different target databases. This allows client applications that use sybase mode to operate independently of the target they are accessing.

When sybase mode is in effect, the access service performs a limited amount of T-SQL syntax transformation on the SQL text that it receives, including text found in the following commands: language, cursor declare, dynamic prepare, and dynamic execute immediate.

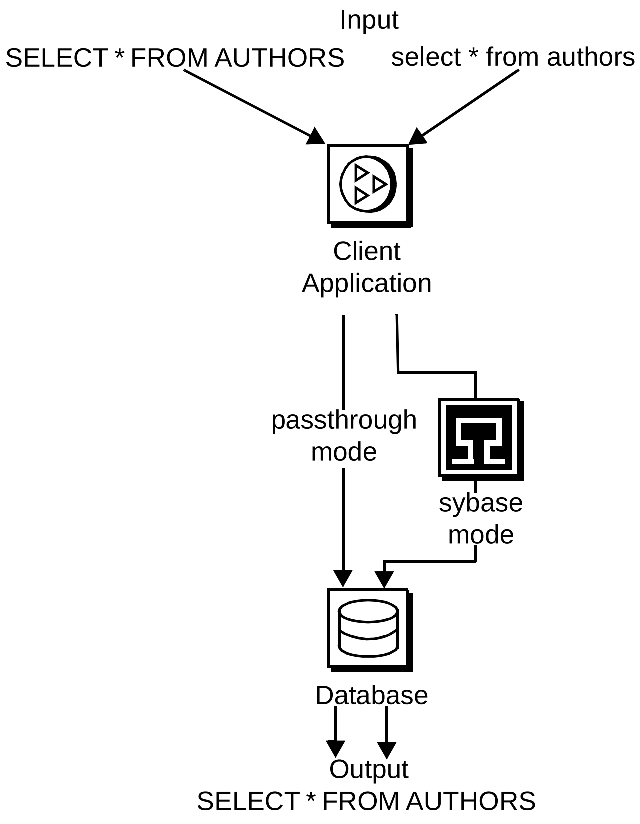
In sybase mode, the access service transforms the SQL text it receives into syntax that the target DBMS supports. If the access service receives syntax that it does not recognize, it passes the text to the DBMS for execution.

Because an application uses this mode for purposes of compatibility with all access services provided by Sybase, it should not issue SQL commands that are unique to any single target DBMS.

Figure 6-1 shows the differences between sybase mode and passthrough mode:

- passthrough mode transfers similar dialect and syntax directly from the client application to the target.
- sybase mode performs translation functions, changing the select statement from lowercase in the client application to uppercase in the target.

Figure 6-1: passthrough and sybase transformation modes



Transformations in sybase mode

The sybase mode makes the following transformations to SQL syntax:

- Provides standard transformations of parameter marker names beginning with the “@” character
- Removes T-SQL comments in the form /*
- Converts T-SQL comparison operators (such as ! and =) into the target DBMS equivalent
- Converts single and double quotation marks used as string delimiters to the appropriate delimiter for the target DBMS
- Strips dollar signs from money constants

Unsupported syntax passes unchanged to the DBMS for processing.

Note The sybase mode does not convert nonquoted tokens to uppercase.

Standard transformations for T-SQL commands

Standard transformations are provided for the commands listed in the following table. Only those portions of the T-SQL syntax that can be directly translated into target DBMS syntax are transformed.

Table 6-1: Standard transformations for T-SQL commands

Command	Description of Transformation
alter table	Adds new columns to an existing table
begin transaction	Begins a new transaction
commit transaction	Commits all work performed for this transaction
create index	Creates a new index on a table
create table	Creates new tables
create view	Creates a new view
delete	Deletes rows from a table
delete (cursor)	Removes rows from a table
delete (dynamic)	Removes rows from a table
drop index	Removes an index from a table
drop table	Removes a table
drop view	Removes one or more views
execute	Runs system or user defined stored procedures.

Command	Description of Transformation
grant	Assigns authorizations to users
insert	Adds new rows to a table or view
insert bulk	(Currently not supported)
prepare transaction	Prepares to commit a transaction
revoke	Revokes permission from users
rollback transaction	Rolls back or aborts the current transaction
select	Retrieves rows from database objects
truncate table	Truncates a table by removing all rows (this statement is not logged and is not part of any transaction)
update	Adds or modifies data in existing rows
update (cursor)	Positional update: changes data in row made current by a read cursor
update (dynamic)	Dynamic update: changes data in existing rows
use	Accesses an existing database

Backward-compatible modes

The DirectConnect access service supports the following three MDI Database Gateway™ transformation modes for backward compatibility:

- tsql0 – works in the same way as passthrough mode and will be phased out eventually.
- tsql1 – provided for backward compatibility only and also will be phased out. It works as it does in the MDI Database Gateway for DB2, version 2.05.
- tsql2 – provided for backward compatibility only. It works the same way as sybase mode.

CHAPTER 7 Issuing RPC Events

This chapter describes how any CT-Library API can generate remote procedure call (RPC) events.

This chapter contains the following topics:

Description of RPCs

The RPC feature allows a stored procedure to initiate an event in a remote database. A client API can invoke an RPC to perform the following:

- Invoke an external stored procedure
- Execute a language statement as an RPC
- Execute a transfer request

Creating and executing Adaptive Server stored procedures

Use the following example to create an Adaptive Server stored procedure that executes an RSP. The new procedure must specify an existing RSP and provide any arguments that the RSP requires.

```
create procedure newcust @custname varchar(nn),
 @custno varchar (nn) as
 begin
 execute servername. . .addcust
 @addname=custname, @addno=custno
end
```

where:

- *@custname* is the customer name to be added.
- *@custno* is the customer number to be added.
- *servername* specifies the access service instance to use. The three periods following *servername* are required.
- addcust is the stored procedure name on the host.
- *@addname* is the stored procedure representing the new customer name.
- *@addno* is the stored procedure representing the new customer number.

To execute the stored procedure in the preceding example using isql, perform the following steps:

1. Connect to Adaptive Server.
2. At the prompt, enter the following:

   ```
   c:>ISQL -Ssybase -Uuser -Ppasswrd
    1> execute newcust xxxx,yyyy
   2> go
   ```

   where:

   - *xxxx* is the new customer name.
   - *yyyy* is the new customer number.

The results obtained depend on the defined addcust stored procedure.

Executing a language statement as an RPC

To execute a SQL language statement to the DirectConnect access service through an RPC, use the following syntax:

```
C:> isql -Sadaptiveserver -Uuser -Ppassword
 1> execute directconnect. . .dcon “select * from
user.authors”
 2> go
```

where:

- *directconnect* is the name of the remote server. The three periods following *directconnect* are required.
- dcon is the keyword of the RPC.

Warning! When using the keyword dcon, the access service will insert a space between each variable.

The access service RPC event handler is sensitive to several RPC keywords, including the keyword dcon used in this example. An RPC can have many parameters. Before the resulting string is executed, all parameters are concatenated. The access service translates the first parameter into a dynamic SQL statement, submits it to the target database, and returns the result set to the client application.

Note For backward compatibility with the MDI Database Gateway™, pcsql is recognized in place of the dcon keyword. For backward compatibility with Net-Gateway™, syrt is recognized in place of the dcon keyword.

The event sequence is as follows:

1. Adaptive Server determines whether the remote server *directconnect* is configured as a remote server:
   - If the remote server is not configured correctly, the request fails.
   - If the remote server is configured correctly, Adaptive Server checks for a site handler connection to the remote server.
2. Adaptive Server does one of the following:
   - If the site handler connection exists, Adaptive Server connects to the remote server, triggering a connect event in DirectConnect.

     If the connect event processes successfully, Adaptive Server triggers an RPC event in DirectConnect and submits the RPC called dcon. The first parameter to the RPC is the dynamic SQL language statement that is executed.
   - If the site handler connection does not exist, Adaptive Server establishes one.

   This connection exists for the term of the RPC. It is reused if Adaptive Server submits other RPCs.

Rules for using language statements as RPCs

Adaptive Server has a strict model for processing language statements as RPC events:

1. It connects to the remote server (DirectConnect) and submits the RPC.
2. After it processes the results, it disconnects.

These quick connects and disconnects provide minimal network traffic.

Additional rules follow in this section.

Validation

When you log on to Adaptive Server, you must use a valid DirectConnect target database user ID and password.

Transformation mode and syntax

All SQL transformation rules apply, including the following:

- If DirectConnect is configured for passthrough mode, the SQL within the double quotes must comply with the target SQL syntax.
- If DirectConnect is configured for sybase mode, the SQL must be in Sybase T-SQL dialect.

Commitment control

The following rules apply:

- If the access service is configured for long transactions, the SQL submitted must not be sensitive to a commit. For example, if the SQL is an insert statement that does not batch a commit into the statement, the insert rolls back using long transaction rules.
- If the access service is configured for short transactions, the SQL submitted is bound by short transactions that supply a commit by default.

Creating a transfer RPC event

The client application can create a stored procedure within Adaptive Server that invokes the DirectConnect access server library transfer function. The access service library receives the transfer command as an RPC event with the following arguments:

- Name of the RPC: "transfer"
- Argument 1: The secondary connection information ({to | from} "server userid pw")
- Argument 2: Either the bulk copy target command or the destination-template sourceselectstatement.
- Argument 3: Either the bulk copy sourceselectstatement or the destination-template sourceselectstatement.

Example of a transfer RPC event

The following example shows a stored procedure that initiates a bulk copy transfer statement. The access service executes the transfer statement against Informix, which receives it as an RPC.

```
create procedure replauth as
 begin
 execute servername. . .transfer
 "to 'servername2 userid password';",
 "with replace into authors;",
 "select * from authors;"
 end
```

where:

- replauth is the name of the stored procedure.
- *servername* specifies the access service to handle the transfer. The three periods following *servername* are required.

  The access service library recognizes anything other than "transfer" in the next position as the name of an ODBC stored procedure.

- *servername2* specifies the secondary database for the transfer command.

Note The RPC can have any number of parameters, since the RPC or parameters are concatenated and executed as a transfer command.

Executing a transfer RPC event

A client can log in to the Adaptive Server on which this procedure is defined and invoke it as follows:

```
execute replauth
```

When Adaptive Server executes replauth, it passes an RPC to the access service. The access service returns any result rows or messages to the client application, not to Adaptive Server.

Using triggers

You can set up any Adaptive Server stored procedure as a trigger that executes automatically when the triggering condition is met.

The following example shows a trigger that calls an RSP named pcrsp when the phone column is updated in the Adaptive Server authors table. In turn, pcrsp updates the authors table on Informix, using au_id to specify the row to update.

```
create trigger updatephone on authors as
 if update (phone)
 begin
 declare @ph varchar(14)
 declare @id varchar(14)
 declare @err int
 select @ph = inserted.phone from inserted
 select @id = inserted.au_id from inserted
 execute servername. . .pcrsp @phone=@ph,
 @au_id=@id
 select @err = @@error
 if (@err >> 0)
 begin
 print ‘error _ rolling back’
 rollback tran
 end
```

```
  else
  commit tran
  end
```

Once it is created, updatephone starts up whenever phone is updated, as shown in the following example:

```
C:>ISQL -Ssybase -Uuser -Ppasswrd
 1> update authors
 2> set phone='xxx-xxx-xxxx'
 3> where au_id like 'yyy-yy-yyyy'
 4> go
```

If the Informix update fails, the DirectConnect access service rolls back the Adaptive Server transaction and shows the following message:

```
@ERR >> 0
```

For more information about RPCs, see the following sources:

- Sybase Open Server *Server-Library/C Reference Manual*
- Sybase Open Client *Client-Library/C Reference Manual*

CHAPTER 8

Understanding the Transfer Process

This chapter describes several concepts of the transfer process and contains the following topics:

Description of the transfer process

The transfer process allows you to transfer rows and columns of data between tables in multiple databases from a client application. Based on your needs and limitations, you can select from one of three transfer options: bulk copy, express, and destination-template.

Description of terms in the transfer process

Following are terms used in the transfer process discussion.

Primary database

This refers to the DBMS that DirectConnect is always connected to; it is implied in the transfer statement.

Secondary database

This database is another DBMS that is defined in the connection string within the transfer statement. The secondary database can be a database that is targeted by another access service.

Source database

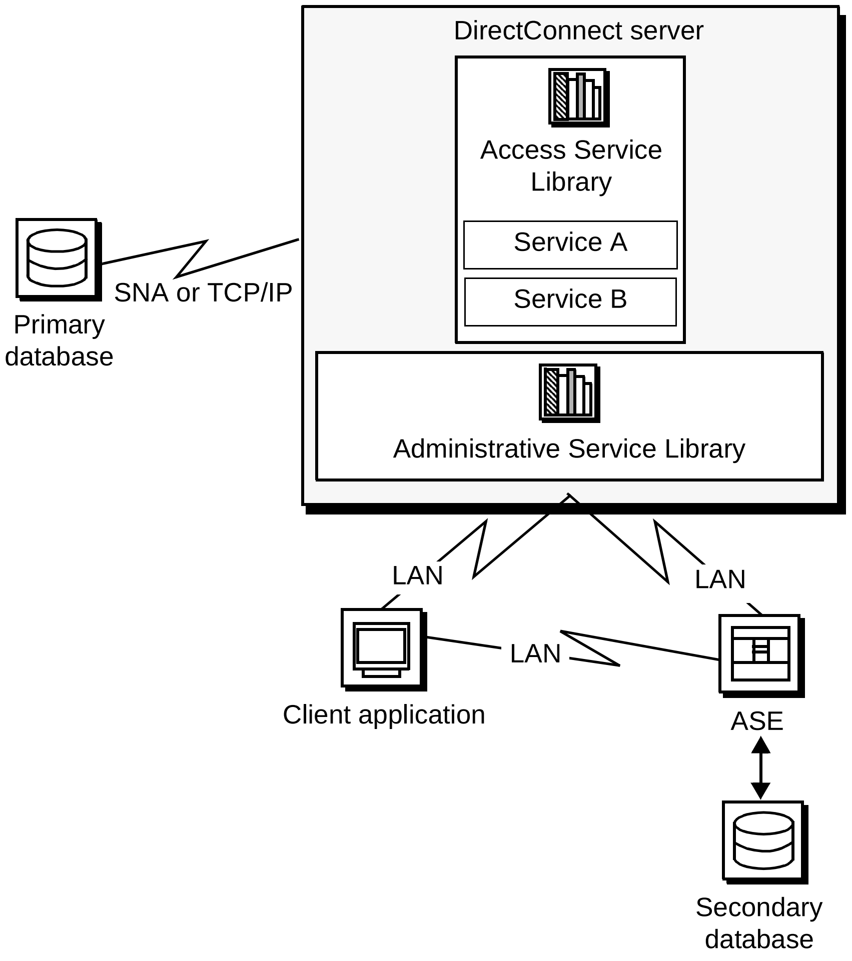
The source is defined as the database the data is coming from.

Destination database

This database is where the data is being sent, also called the *target*. Either the primary or secondary database can be the source or destination, depending on whether you use the transfer from or transfer to command.

The following figure shows the access service configured to access a particular primary database. The secondary database is an Adaptive Server database located on the LAN. The client application can transfer data from the primary database to Adaptive Server, or from Adaptive Server to the primary database.

Figure 8-1: Access service data transfer between databases.



During a transfer:

- Data flows from a table in the source database, through DirectConnect, to the target database. Although the client application initiates the transfer, the data does *not* flow through it.
- The DirectConnect server becomes a client to the secondary database.

Description of bulk copy, express, and destination-template transfer

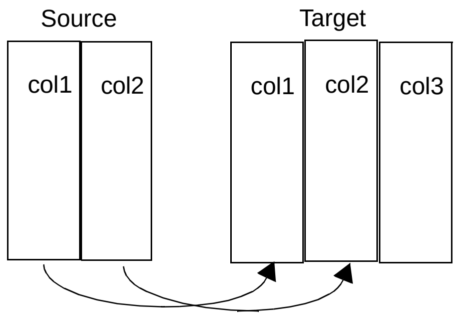
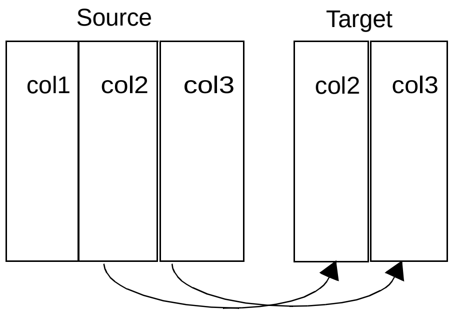
The following describes how bulk copy transfer, express transfer, and destination-template transfer process data:

- Bulk copy transfer and express transfer allow you to quickly and directly transfer large amounts of compatible data between databases.
- Destination-template transfer allows you the flexibility and control to insert the data in a template before sending it to the target database.

The following table further describes the conditions that determine the type of transfer you select.

Table 8-1: Comparison of two transfer command types

Use bulk copy and express transfer to:	**Use destination-template transfer to:**
Execute the transfer quickly	Exercise more control over the transfer
Perform implicit datatype conversions	Move tables of data in which you need to explicitly specify datatype conversion, such as when the data is structurally incompatible
Move entire tables of data in which column types are compatible between the source and destination databases:	Perform actions other than INSERT against a target database using input from the source database, such as UPDATE, DELETE, and CREATE


transfer to secondary_connection;
with replace into target_table;
select col1, col 2 from source_table
	
	transfer to secondary_connection;
	select col2, col3 from source_table;
	insert into target_table values (?,?)

For detailed information about bulk copy and express transfer, see Chapter 9, “Using Bulk Copy Transfer and Express Transfer.” For detailed information about
destination-template transfer, see Chapter 10, “Using Destination-Template Transfer.”

Transfer direction

You can transfer data in two directions:

- A transfer to statement transfers data from the *primary* database to the *secondary* database. The primary database becomes the source database to the secondary database, which is the target.
- A transfer from statement transfers data from the *secondary* database to the *primary* database. The secondary database becomes the source database to the primary database, which is the target.

For example, when you execute a bulk copy transfer from statement, the secondary database (either Adaptive Server or another database) is the source of the data to be transferred. The primary database becomes the destination database, or target.

Note For the implications of using one transfer direction over another, see "Datatype conversion for transfer processing" on page 97.

Transfer compatibility

Sybase recommends that you use the transfer from command from the primary database when you transfer data between two access services. Using transfer from guarantees native datatype mapping and returns the proper datatype result set.

Unit of work as defined in the transfer process

A unit of work is one or more requests that execute, commit, or roll back as a group. Following are descriptions of a unit of work for bulk copy, express, and destination-template transfer.

Bulk copy and express transfer

Unit of work is based on the setting of the BulkCommitCount property:

- If BulkCommitCount is set to 0, the entire transfer is treated as a unit of work. The access service performs a commit after the last row of data is inserted into the target table, even if value errors occurred for individual rows of the transfer.

- If BulkCommitCount is set to a non-zero value, each block of BulkCommitCount rows is treated as a unit of work. The access service issues a commit after each block of BulkCommitCount rows. For example, if BulkCommitCount is set to 50, each block of 50 rows is treated as a unit of work, and a commit is issued after each 50 rows.

For information about value errors, see the section “Value errors” on page 98.

Destination-template transfer

When a destination-template transfer statement moves data from Adaptive Server to the primary database, the access service automatically sets the StopCondition property to none. Subsequent commit and rollback processing is determined by whether short or long transactions are in effect:

- If short transactions are in effect, the access service issues a commit after each batch, whether or not errors occurred in the request. In this case, each batch of inserts is a unit of work.
- If long transactions are in effect, the access service issues a commit at the end of the entire transfer. Because StopCondition is set to none, the access service never issues a rollback. In this case, the entire transfer is a unit of work.

Transfer targets

The transfer statement allows you to move data in either direction between the primary database and:

- Adaptive Server
- ASE/CIS
- Other access service and legacy products or services:
  - DB2 UDB (z/OS, UNIX, and Windows)
  - Informix
  - Microsoft SQL Server
  - Any other supported database, or ODBC accessible database
- Any Open Server application that supports SQL

Datatype conversion for transfer processing

Datatype conversion is handled differently for the transfer types. After converting the incoming source database datatypes, the access service does one of the following:

- For bulk copy and express transfer processing, the source datatypes are converted into the actual datatypes of the target columns. If the source and target columns have incompatible datatypes, the transfer ends with an error.
- For destination-template transfer processing, the datatype qualifiers specified are used, with the question marks in the template. When the question marks do not have qualifiers, the access service uses the datatypes of the source to determine the default qualifiers.

Warning! DirectConnect cannot correctly transfer varchar values containing empty strings (zero length non-null strings), in the bulk copy and destination-template transfer process. Empty string varchar values are interpreted as *NULL* values. Express transfer processes this empty string correctly.

Transfer errors and error handling

Transfer processing errors can occur due to:

- Structural errors, for which the access service cancels the transfer process *before* any rows are transferred
- Value errors, which occur on a row-by-row basis *during* the transfer process

Each of these transfer error types is explained in the following sections.

Structural errors

A structural error can be one of two types:

- Incompatible datatypes
- An incompatible number of columns

Each type is described in the following sections.

Incompatible datatypes

Datatypes are incompatible when source and target table datatypes cannot be mapped to one another. For example, binary to datetime transfers are not allowed.

- In bulk copy and express transfers, before the access service moves any data from the source database to the target database, the access service compares datatypes in the source to datatypes in the target.
- In destination-template transfers, the access service compares the source table datatypes to the qualifiers set in the destination-template statement.

For all transfer types, if any columns have incompatible datatypes, the access service cancels the transfer without attempting to transfer any rows.

Incompatible number of columns

When the access service detects an unequal number of columns between the source and the target, a structural error can occur.

- For bulk copy and express transfer:
  - If the number of source columns exceeds the number of target columns, the access service cancels the transfer.
  - If the number of target columns exceeds the number of source columns, processing continues if all of the extra columns of the target table accept nulls. If any one of the extra columns in the destination table does not accept nulls, the access service cancels the transfer.
- For destination-template transfer:
  - If the number of columns returned by the sourceselectstatement exceeds the number of question marks in the destination-template transfer statement, the access service cancels the transfer.
  - If the number of question marks in the destination-template transfer statement exceeds the number of source columns, the keyword null replaces the extra question marks for each row of the transfer.

Value errors

Value errors occur during bulk copy transfer or destination-template transfer processing when the value being inserted has one of the following characteristics:

- It cannot be converted to the target column’s datatype.

- It is out of range for the target column’s datatype.

Note Express transfer does not support value error handling: If an error occurs, it aborts the entire transfer.

The access service handles these errors using the following properties:

- CharConvertError
- NumConvertError
- DatetimeConvertError
- DefaultDate
- DefaultTime
- DefaultNum

If the SendWarningMessages property is set to yes, the access service sends a message to the client application when it encounters value errors.

In addition, the preceding properties can be used to fill in default values when datatype conversion fails during bulk copy and destination-template transfer.

For more information about configuration properties, see Chapter 2, “Configuring the Access Service Library for DirectConnect.”

For information about values that cause errors for database access service during bulk copy transfer, see Chapter 9, “Using Bulk Copy Transfer and Express Transfer.”

Error reporting for transfer processing

You can use one of three methods to obtain error information about bulk copy and destination-template transfer processing:

- Include the with report phrase in the transfer statement. When you include this phrase, the access service returns a result set containing one VARCHAR column and one row that indicates the number of rows transferred, rejected, and modified during processing.
- Immediately following a transfer process, execute the following:

  ```
  select @@RejectedRowCount
  
  select @@DefaultedRowCount
  ```

These global variables return the number of rejected or defaulted rows.

- Set SendWarningMessages to yes, so that the access service returns warning messages to the client when data conversion errors occur.

Controlling processing with the TransferErrorCount property

During bulk copy and destination-template transfer processing, the access service automatically sets the StopCondition property to none. Then, it uses the value set in the TransferErrorCount property to determine how many error rows are allowed before processing stops. You can set this value with the following statement:

```
set TransferErrorCount nnn
```

The default setting is 0 (zero), which causes the access service to ignore errors.

CHAPTER 9

Using Bulk Copy Transfer and Express Transfer

This chapter describes bulk copy and express transfer. It contains the following topics:

Overview

Bulk copy and express transfer initiate a direct transfer of data between two databases from the client application. You use a bulk copy or express transfer statement to copy large amounts of data between similar tables. The following table describes the conditions that determine which type of transfer you select.

Table 9-1: When to use bulk copy and express transfer

Use bulk copy transfer:	Use express transfer:
When the source or destination database is ASE and does not use an ASE ODBC driver.	When you use ODBC drivers for both the source and destination database.
To exercise more control over transfer.	To execute the transfer quickly.
When you require diverse datatype conversions.	When you do not require any datatype conversions.

Limitations for bulk copy and express transfer

For bulk copy and express transfer, the following limitations apply:

- The transfer statement must be the only statement in a request.
- The table (the target) into which you want to transfer data must already exist, because the transfer statement does not create new tables.
- The structure of the target table must match the structure of the source table.
- For bulk copy and express transfer to work, the secondaryname to the secondary database must be recorded in the following DirectConnect files:
  - For bulk copy transfer, the *interfaces* file for UNIX, or in the *sql.ini* file for Windows.
  - For express transfer, the secondaryname must match a Data Source Name (DSN) in the *ODBC system information* file.
- Unicode datatypes are not supported.
- The first 32K of long character and long binary values are supported. Transfer processing truncates longer values without any warning.

Transfer process

The following describes the bulk copy and express transfer process.

❖ **To initiate the bulk copy and express transfer process**

1. The client application initiates a transfer request.
2. The access service receives the transfer request and executes the sourceselectstatement against the source database to retrieve the schema of the result set, including column datatypes, length, precision, and scale.
3. The access service queries the target table for a description of the target table columns and compares this information to the structure of the result set for the following criteria:
   - The target table must have at least as many columns as the result set.
   - The datatype of each result set column must be able to be converted to the datatype of the target column.

   If either of these tests fails, the access service stops processing the transfer and issues an error message.

4. If the transfer statement includes the with replace or truncate clause, the access service deletes data in the target table, provided the user ID of the person executing the request is authorized to do so. If the user ID is not authorized, the transfer fails.
5. The access service maps the columns from the result set to the columns in the target table in the same order. The access service attempts to insert NULL values (if allowable) in all columns in the target table that does not have corresponding columns in the result set.
6. The access service prepares an insert or equivalent bulk load statement for execution against the target table.
7. For bulk copy transfer only, if conversion errors occur as rows are inserted (for example, a value is out of range), the invalid rows are handled according to the values set in the following properties:
   - CharConvertError
   - NumConvertError
   - DatetimeConvertError
   - DefaultDate
   - DefaultTime
   - DefaultNum

   The transfer continues processing. If the SendWarningMessages property is set to yes, the access service sends a warning message to the client application.

Syntax

The following is the required syntax for a bulk copy and express transfer statement:

> transfer [with report]
>
> {to | from} ‘secondaryname userid password’;
>
> with {insert | replace | truncate} into *tablename*;
>
> *sourceselectstatement*

where:

- transfer must begin all transfer statements.

- with report is an optional phrase specified in the first line of the transfer statement. It instructs the access service to return processing information to the client application.

  This information is returned as a result set consisting of a VARCHAR column and a single row. The row contains the number of rows transferred, rejected, and modified during processing.

- {to | from} indicates the direction of the transfer:
  - to specifies that the data is transferred from the primary database to the secondary database.
  - from specifies that the data is transferred from the secondary database to the primary database.

- secondaryname userid password is a three-part character string that provides the information needed to connect to the secondary database:
  - secondaryname is the name used to identify the secondary database and must be recorded in the following DirectConnect files:
    - For bulk copy transfer, in the *interfaces* file for UNIX, or in the *sql.ini* file for Windows.
    - For express transfer, the secondaryname must match a data source name (DSN) in the *ODBC system information* file.
  - userid and password must be valid on the secondary database. If the password is NULL, you can substitute an asterisk for password and it will be corrected to a NULL when sent to the secondary connection. Exactly three tokens are sent to the secondary connection.

  All of the elements of the character string must be enclosed in single or double quotes in the order shown.

- with {insert | replace | truncate} into specifies whether the data is appended onto the target table (insert) or the existing data is deleted and replaced (replace or truncate).

  When transferring data to Adaptive Server, the truncate option causes transfer to issue a truncate, rather than a delete against the target table. For other target databases, delete and truncate are equivalent.

- *tablename* specifies the table into which data is inserted or replaced. The table must already exist because the transfer statement does not create a new one in the target database.

- sourceselectstatement specifies a SQL statement that is executed against the source database to produce the result set used in the transfer.

  This statement can be any statement the source database will accept, including stored procedures. SQL transformation is not performed on the sourceselectstatement. It must be in the source database SQL dialect.

Express transfer

To improve performance transferring bulk data between data sources, Sybase has a bulk copy transfer called "express transfer." It transfers data faster than bulk copy transfer using ODBC bulk APIs. Because the new express transfer feature uses the same syntax as the DirectConnect bulk copy transfer, you can take advantage of express transfer without modifying your applications.

Availability

This new feature is available for the following ECDA products:

- DirectConnect DB2 UDB
- DirectConnect for Informix
- DirectConnect for Microsoft SQL Server
- DirectConnect for ODBC

To use this feature, you will need one of these products and an ODBC driver from another DirectConnect or Adaptive Server Enterprise (ASE) product. For example, to transfer data between Microsoft SQL Server and ASE requires DirectConnect for Microsoft SQL Server and an ASE ODBC driver.

Functional differences between bulk copy and express transfer

There are functional differences in datatype conversion and error handling between bulk copy and express transfer:

- Datatype conversion
  - In bulk copy transfer, data being transferred is converted into intermediate Open Server datatypes.
  - In express transfer, the ODBC driver converts the datatypes automatically, and no conversion takes place in DirectConnect.
- Error handling
  - If a datatype conversion error occurs, bulk copy transfer supports error handling as defined in the following section "Bulk copy and express transfer errors" on page 110.

- Express transfer does not support error handling: If an error occurs, it aborts the entire transfer.

Using express transfer

❖ To prepare to use express transfer

1. Set the TransferExpress property to yes, which causes the bulk copy transfer statements to be interpreted as express transfer statements.
2. Enter the ODBC data source name as the secondary name in the transfer statement.

Certified express transfer targets

Express transfer is now certified for the following target combinations:

- DB2 UDB OS/390, to and from Microsoft SQL Server 2000, requires the following:
    - DirectConnect for Microsoft SQL Server
    - DirectConnect for DB2 UDB driver
- UDB DB2 OS/390, to and from ASE, requires the following:
    - DirectConnect for DB2 UDB
    - Sybase ASE ODBC driver
- Microsoft SQL Server 2000, to and from ASE, requires the following:
    - DirectConnect for Microsoft SQL Server
    - Sybase ASE ODBC driver

To use express transfer with ASE, a Sybase-supplied ASE ODBC driver is available on the Windows platforms. For DirectConnect on UNIX platforms, a UNIX ASE driver must be purchased from Data Direct Technologies. Additional information regarding a driver can be found at www.datadirect.com.

Note When you use an ASE driver, make the ASE connection the secondary connection. Use the ASE DSN for the secondary name.

An express transfer example

This example describes how you can express transfer data from DB2 to Microsoft SQL Server. It uses an isql connection to DB2 through DirectConnect for DB2 UDB (primary server), and an ODBC DSN (on the primary server machine) for Microsoft SQL Server called MSQL-DSN.

❖ To use express transfer from DB2 to Microsoft SQL Server

1. Connect to DB2 through the primary server by entering the following:

```
-isql -S dcdb2udb - Uuserid -Ppassword
```

2 Set TransferExpress to yes.

3 Enter the following:

```
Transfer to 'MSQL-DSN userid password';
with insert into MSQL-table;
Select * from db2-table
```

Datatype conversion for express transfer statements

The following table shows the acceptable source datatypes that the access service can convert into corresponding target destination datatypes.

Table 9-2: Datatype conversions

	Target datatypes				
Source datatypes	**CHAR, VARCHAR, LONGVARCHAR, TEXT (CLOB),**	**BIGINT, DECIMAL, DOUBLE FLOAT, INTEGER, MONEY, MONEY4, NUMERIC, REAL, SMALLINT, TINYINT**	**DATETIME, DATETIME4, TIMESTAMP**	**BINARY, VARBINARY, LONGVARBINARY, IMAGE (BLOB)**	**BIT**
CHAR, VARCHAR, LONGVARCHAR, TEXT (CLOB)	X	X	X	X	X
BIGINT, DECIMAL, DOUBLE, FLOAT, INTEGER, MONEY, MONEY4, NUMERIC, REAL, SMALLINT, TINYINT	X	X			
BINARY, VARBINARY, LONGVARBINARY, IMAGE (BLOB)	X			X	

Source datatypes	Target datatypes				
	CHAR, VARCHAR, LONGVARCHAR, TEXT (CLOB),	**BIGINT, DECIMAL, DOUBLE FLOAT, INTEGER, MONEY, MONEY4, NUMERIC, REAL, SMALLINT, TINYINT**	**DATETIME, DATETIME4, TIMESTAMP**	**BINARY, VARBINARY, LONGVARBINARY, IMAGE (BLOB)**	**BIT**
DATETIME, DATETIME4, TIMESTAMP	X		X		
BIT	X	X			X

Datatypes resulting from this conversion are converted into actual datatypes in the target column, as shown in Table 9-2. If a column match is incompatible, the transfer ends with an error. If data conversion errors occur, you should try swapping the connections to the drivers used for the primary server and the secondary server.

Note The datatype conversions identified in Table 9-1 may not be available for express transfer. It varies and depends on the ODBC driver that is used.

Processing bulk copy values

The following guidelines apply to character, numeric, date, and binary datatype values.

Warning! DirectConnect cannot correctly transfer varchar values containing empty strings (zero length non-null strings), in the bulk copy transfer process. Empty string varchar values are interpreted as *NULL* values.

Character datatypes

Character datatypes (CHAR, VARCHAR, TEXT) can be converted to any other datatype. Conversely, every datatype can be converted to character data. You must verify that the character string is able to be converted to the target datatype.

For example, the character string “450” can be converted to a numeric datatype such as *INTEGER* or *DECIMAL*, but the character string “Hello” causes a value error going to a numeric datatype.

Numeric datatypes

Numeric datatypes can be converted to other numeric datatypes or to character datatypes, but they cannot be converted to binary or date datatypes.

Additional guidelines are as follows:

- All numeric conversions use rounding.
- Any loss of digits to the left of the decimal results in an error. For example, an integer of value 123 cannot be converted to a decimal (4,2) value without losing a digit to the left of the decimal point.
- Any loss of digits to the right of the decimal point is not considered an error. For example, a float of value 123.456 is converted to an integer of value 123 without an error.
- When you transfer data from a decimal column to a float column, the precision of the result is not better than the precision of the target column.

  For example, if you transfer data from a decimal(15,0) column to a float column, then back to a decimal(15,0) column, the results in the target decimal(15,0) column do not match the results of the source decimal(15,0) column, due to the float column precision.

Date datatypes

Date datatypes can be converted to other date datatypes or to character strings. However, they cannot be converted to numeric or binary datatypes.

Binary datatypes

Binary datatypes can be converted to other binary datatypes or to character datatypes. However, they cannot be converted to numeric or date datatypes.

Bulk copy and express transfer errors

Bulk copy transfer errors and express transfer errors are handled differently.

Express transfer errors

Express transfer does not support errors regarding individual rows. If an error occurs, the entire transfer is aborted. Any value errors that occur are handled by the ODBC driver.

Bulk copy transfer errors

Bulk copy transfer errors are handled by the access service. Value errors occur during transfer processing when the value being inserted is out of range for the column datatype.

Bulk copy value processing rules

The following rules apply for values when using the bulk copy transfer process:

- NULL values:

  If a source column contains NULL values but the destination does not allow them, the row is rejected.

- Numeric data:

  - All numeric conversions use rounding.
  - Any loss of digits to the left of the decimal results in an error. For example, an integer of value *123* could not be converted to a decimal(4,2) value without losing a digit to the left of the decimal point.
  - Any loss of digits to the right of the decimal point is not considered an error. For example, a float of value *123.456* would be converted to an integer value of *123* without error.

- Binary data:

  Binary data is transferred to a binary column without byte translation. A byte value in the source will have the same value in the destination.

Values that cause errors

The following table shows data values that can cause errors during bulk copy transfer.

Table 9-3: Values that cause errors

Source input datatypes	Target (destination) datatypes	Error conditions for bulk copy
char, varchar, text	char, varchar, text, binary, varbinary, or image, LONGVARCHAR	Source data is longer than the destination column.
char, varchar, text	decimal, DECIMAL	• Source is not a valid decimal string (must contain an optional leading sign and decimal digits). • Number of digits to the left of the decimal point is greater than the destination column precision minus scale. Any digits to the right of the decimal will be lost without error.
char, varchar, text	integer	• Source is not a valid decimal string (must contain an optional leading sign, decimal digits, decimal point, and fractional decimal digits). • String of digits to the left of the decimal point is greater than 2147483647 (positive values) or less than -2147483648 (negative values). **Note** Any digits to the right of the decimal will be lost without an error being generated.
char, varchar, text	smallint	• Source is not a valid decimal string (must contain an optional leading sign, decimal digits, decimal point, and fractional decimal digits). • String of digits to the left of the decimal point is greater than 32767 (positive values) or less than -32768 (negative values).
char, varchar, text	tinyint	• Source is not a valid positive integer string (must contain an optional leading "+" and decimal digits). • String of digits does not form an integer value between 0 and 255.
char, varchar, text	OS bit	Source data length is greater than 1 or source value !="0" or "1."
char, varchar, text	float. DOUBLE	Source is not a valid floating point format string (must contain optional leading sign, decimal digits, optional decimal point, fractional decimal digits, and optional *E[+|-] nnn* exponent).

Source input datatypes	Target (destination) datatypes	Error conditions for bulk copy
char, varchar, text	real	• Source is not a valid floating point format string (must contain optional leading sign, decimal digits, optional decimal point, fractional decimal digits, and optional *E[+|-] nnn* exponent). • Target (destination) is Adaptive Server, and the value is greater than 3.402823466E38 or less than -3.402823466E38.
char, varchar, text	date	Source is not an ISO format date (*YYY-MM-DD*) or a valid Adaptive Server date/time string with the year later than 1753.
char, varchar, text	time	Source is not an ISO format time (*HH.MM.SS*) or an *HH:MM:SS* format time.
char, varchar, text	ODBC TIMESTAMP	Source is not an ISO format date (*YYYY-MM-DD-HH.MM.SS*) or a *YYYY-MM-DD-HH.MM.SS.NNNNNN* date with *YYYY* greater than 0001, or a valid Adaptive Server date/time string with the year later than 1753.
char, varchar, text	datetime	Source is not an ISO format date, time, or timestamp with a year later than 1753 or a valid Adaptive Server date/time string with a year later than 1753.
char, varchar, text	datetime4	Source is not an ISO format date, time, or timestamp with a year later than 1899 and the year, month, and day earlier than Jun 7, 2079, or a valid Adaptive Server date/time string with a year later than 1899 and the year, month, and day earlier than Jun 7, 2079.
char, varchar, text	money	Source is not a valid decimal string (must contain an optional leading sign, decimal digits, optional decimal point, and fractional decimal digits), and the value is greater than 922337203685477.5807 or less than -922337203685477.5808.
char, varchar, text	money4	Source is not a valid decimal string (must contain an optional leading sign, decimal digits, optional decimal point, and fractional decimal digits), and the value is greater than 214748.3647 or less than -214748.3648.
binary, varbinary, image	char, varchar, text, binary, varbinary, image, LONGVARCHAR	Source data is longer than the destination column.
byte, int, smallint	char, varchar, text	Destination column is too small to hold the digits required to express the value. For example, the source value is 103 and the destination column is char(2).
byte, int, smallint	bit	Source value !=0 or 1.

Source input datatypes	Target (destination) datatypes	Error conditions for bulk copy
smallint, int, float, real, money, money4, decimal	decimal	Destination column precision minus scale is too small to hold the value. For example, a source data value of 98 requires destination column precision minus scale of 2.
money, money4, decimal, numeric	decimal	If precision=scale=maximum, precision for the datatype does not transfer properly when the data value is 0. A workaround is to alter the table to avoid one of these conditions.
smallint	tinyint	Source value is greater than 255 or less than 0.
int	smallint	Source is greater than 32767 or less than -32768.
int	money4	Source is greater than 214748 or less than -214748.
int	tinyint	Source is greater than 255 or less than 0.
bit	decimal	Target (destination) column precision minus scale is less than 1.
float, real	char, varchar, text	Target (destination) column is too small to hold the digits required to express the value. For example, source is 1030303E+30 and destination column is char(12).
float, real, money	int	Source value greater than 2147483647.0 or less than -2147483648.0.
float, real, money, money4, decimal	smallint	Source is greater than 32767.0 or less than -32768.0.
float, real, money, money4, decimal	tinyint	Source is greater than 255.0 or less than 0.0.
float, real	money	Source value is greater than 922337203685477.0 or less than -922337203685477.0. • Since float accuracy is 15 digits, a value of this magnitude is accurate only to the nearest dollar. • Since real accuracy is 7 digits, a value of this magnitude is accurate only to the nearest hundred million dollars.
float, real, money, decimal	money4	Source is greater than 214748.3647 or less than -214748.3648.
float, real, money, money4, decimal	bit	Source value !=0.0 or 1.1.
money, money4, decimal	char, varchar, text	Target (destination) column is too small to hold digits required to express value. For example, source is 100000000.001 and destination column is char(12).
datetime, datetime4	char, varchar, text	Target (destination0 column length is less than 19.
datetime	datetime4	Date portion of source value is earlier than Jan 1 1900 or later than Jun 6 2079.

Bulk copy transfer error reporting

You can obtain bulk copy transfer error information in the following ways:

- If you include with report in the transfer statement, you receive a result set containing one VARCHAR column and one row indicating the number of rows transferred, rejected, and modified during processing.
- You can execute @@RejectedRowCount or @@DefaultedRowCount immediately after a successful transfer. These global variables return the number of rejected or defaulted rows.
- If you set SendWarningMessages to yes, the access service returns data conversion errors to the client application.

CHAPTER 10

Using Destination-Template Transfer

This chapter describes the destination-template transfer process. It contains the following topics:

Overview

The destination-template transfer statement allows you to transfer data and also allows you to feed a result set into a template and force the template to perform the following operations:

- Create a full-image copy (insert)
- Create an incremental copy (insert)
- Modify column values (update or delete)
- Perform structural modifications (create or alter)
- Change database permissions (grant or revoke)
- Execute remote stored procedures (execute or use)
- Execute other arbitrary SQL statements

Description of destination-template transfer processing

During a destination-template transfer, the access service inserts the data values it retrieves with the sourceselectstatement from the source database into the destination-template SQL clause. This clause contains one question mark for each column in the result set of the sourceselectstatement.

Each value from the result set is substituted for the corresponding question mark in the template on a row-by-row basis. The access service executes the resulting statement against the target database.

When you use the destination-template transfer statement, the following restrictions apply:

- The transfer statement must be the only statement in a request.
- The table into which you want to transfer data (the target) must already exist. The transfer statement does not create new tables in the transfer target.
- The structure of the target table must match the structure of the source table.
- For any transfer to work, the ConnectionSpec to the secondary connection must be recorded in the *interfaces* file for DirectConnect installations on UNIX or in the *sql.ini* file for DirectConnect installations on Windows.

Warning! DirectConnect cannot correctly transfer varchar values containing empty strings (zero length non-null strings), in the destination-template transfer process. Empty string varchar values are interpreted as *NULL* values.

Syntax

The syntax for a destination-template transfer statement is as follows:

transfer [with report]

{to | from} '*secondaryname userid password*';

sourceselectstatement;

destinationtemplatestatement

where:

- transfer must begin all transfer statements.
- with report is an optional phrase specified in the first line of the transfer statement that instructs the access service to return processing information to the client application.

  This information is returned as a result set that consists of one VARCHAR column and one row. The row contains the number of rows transferred, rejected, or modified during processing.
- {to | from} indicates the direction of the transfer:
  - to specifies that the data is transferred from the primary database to the secondary database.
  - from specifies that the data is transferred from the secondary database to the primary database.
- *‘secondaryname userid password’* is a character string that provides the information needed to connect to the secondary database:
  - *secondaryname* is the name used to identify the secondary database.
  - *userid* and *password* must be valid on the secondary database. You can substitute an asterisk for *password*, if a password is not specified.

  All of the elements of the character string must be enclosed in single or double quotes in the order shown.
- sourceselectstatement specifies a SQL statement that is executed against the source database to produce the result set that will be used in the transfer. This statement can be of any complexity acceptable to the source database, including stored procedures. SQL transformation is not performed on the sourceselectstatement. The transformation must be in the source database SQL syntax.
- destinationtemplatestatement is a SQL statement or any statement that is valid for the target database environment where it executes. SQL transformation is not performed on the destinationtemplatestatement. The transformation must be in the destination database SQL syntax.

  This statement can include question marks as placeholders for the data values that will be inserted. It can also include qualifiers to specify datatypes for the question mark placeholders in the destinationtemplatestatement.

To increase processing efficiency, you can batch destination-templates together for processing. Use the TransferBatch configuration property or a set statement to specify how many templates to batch.

Datatype qualifiers

Qualifiers tell the access service how to format data that is inserted for a placeholder. If you do not supply a qualifier, the access service applies default transformations.

Qualification is required for date and time values. You can use the ?T, ?t, ?D, and ?d qualifiers for dates, or you can create a custom qualifier using special qualifiers. For information about special qualifiers, see "Special date and time qualifiers" on page 121.

The following table defines valid datatype qualifiers.

Table 10-1: Destination-template transfer datatype qualifiers

Placeholder/ Qualifier	Definition
?C	Character string enclosed in quotes
?N	Numeric data, no quotes
?D	Standard format Adaptive Server datetime data enclosed in quotes
?T	Standard format ISO TIMESTAMP data enclosed in quotes, *'YYYY-MM-DD-hh.mm.ss.nnnnnn'*
?d	*'mm/dd/yyyy'*
?t	*'hh:mm:ss'*
?X	Standard format Adaptive Server hexadecimal data (for example, 0xffee) used for transferring binary data
?x	Standard format hexadecimal data (for example, X'FFEE') used for transferring binary data
?y	*'yy/mm/dd'*
?G	G'<...>' used for transferring graphic datatypes or formatting a graphic constant from binary character data
?g	GX'<...>' used for transferring graphic datatypes or formatting a graphic constant from binary character data (returns data in hexadecimal format)

The following three tables show the effects of qualifiers on datatypes.

Note For each table, special circumstances are detailed in the text following the table.

The first table shows the effects of the ?C, ?N, ?D, and ?T qualifiers.

Table 10-2: Effects of qualifiers on datatypes (1)

Open Server datatype	**Default**	**Effects (by qualifier)**			
		?C	**?N**	**?D**	**?T**
CS_CHAR CS_VARCHAR CS_TEXT	?C	Quote	No quote	Convert to Open Server datetime string, quote	Convert to ISO TIMESTAMP, quote
CS_BIT, CS_INT1 CS_INT2 CS_ INT4 CS_REAL CS_FLOAT	?N	Convert to char, quote	Convert to char, no quote	n/a	n/a
CS_MONEY CS_MONEY4 CS_DECIMAL	?N	Convert to char, quote	Convert to char, no quote	n/a	n/a
CS_DATETIME CS_DATETIME4	?D	*'MON DD YYYY hh:mm'* [AM or PM]	n/a	*'MON DD YYYY hh:mm:ss:nnn'*	*'YYYY-MM-DD-hh.mm.ss. nnnnnn'*

For CS_CHAR, CS_VARCHAR, AND CS_TEXT used with the ?D qualifier:

- If the source is an ISO TIMESTAMP, it is converted to *'Mon dd yyyy hh:mm:ss:nnn'*.
- If it is an ISO DATE, it is converted to *'Mon dd yy'*.
- If it is an ISO TIME, it is converted to *'Mon dd yy hh:mm:ss'* using the value from the DefaultDate property as the date portion of the value.

For CS_CHAR, CS_VARCHAR, AND CS_TEXT used with the ?T qualifier:

- If the source is an ISO DATE or TIME, the DefaultDate and DefaultTime property values are used to fill in missing information.

The following table shows the effects of the ?y, ?d, ?t, and ?x qualifiers.

Table 10-3: Effects of qualifiers on datatypes (2)

Open Server datatype	**Default**	**Effects (by qualifier)**			
		?y	**?d**	**?t**	**?x**
CS_CHAR CS_VARCHAR CS_TEXT	?C	Convert and Quote	Convert and Quote	Convert and Quote	Convert to hex; leading X’ trailing ’. For example X’ab70’
CS_BIT, CS_INT1 CS_INT2 CS_INT4 CS_REAL CS_FLOAT CS_MONEY CS_MONEY4 CS_DECIMAL	?N	n/a	n/a	n/a	n/a
CS_DATETIME CS_DATETIME4	?D	*’yy/mm/dd’*	*‘mm/dd/yyyy’*	*‘hh:mm:ss’*	n/a
CS_BINARY CS_VARBINARY CS_IMAGE	?X or ?x	n/a	n/a	n/a	Convert to hex; leading X’ trailing ’. For example X’ab70’

For CS_CHAR, CS_VARCHAR, AND CS_TEXT used with the ?y qualifier:

- If the source is an ISO DATE, TIME, or TIMESTAMP, it is converted to *‘yy/mm/dd*’.

For CS_CHAR, CS_VARCHAR, AND CS_TEXT used with the ?d qualifier:

- If the source is an ISO DATE, TIME, or TIMESTAMP, it is converted to *‘mm/dd/yy’*.

For CS_CHAR, CS_VARCHAR, AND CS_TEXT used with the ?t qualifier:

- If the source is an ISO DATE, TIME, or TIMESTAMP, it is converted to *‘hh:mm:ss’*.

For all datatypes used with the ?x qualifier:

- If the target database is ODBC , ?x converts the data to the standard ODBC hexadecimal format (a quoted hexadecimal number with a leading X).

The following table shows the effects of the ?X and ?O qualifiers.

Table 10-4: Effects of qualifiers on datatypes (3)

Open Server datatype	Default	Effects (by qualifier)
		?X
CS_CHAR CS_VARCHAR CS_TEXT		Convert to hex; leading 0x, no quote
CS_BIT, CS_ INT1 CS_INT2 CS_INT4 CS_REAL CS_FLOAT CS_MONEY CS_MONEY4 CS_DECIMAL	?N	n/a
CS_DATETIME CS_DATETIME4	?D	n/a
CS_BINARY CS_VARBINARY CS_IMAGE	?X or ?x	Convert to hex; leading 0x, no quote

For CS_BINARY, CS_VARBINARY, AND CS_IMAGE datatypes used with the ?X qualifier:

- If the target database is ODBC, ?x converts the data to the standard ODBC hexadecimal format (a quoted hexadecimal number with a leading X).

Special date and time qualifiers

You can combine special date and time qualifiers and construct the date or time format that the target database requires, using the following rules:

- Enter special characters in either uppercase or lowercase.
- Separate special characters by any arbitrary character, such as a hyphen, slash, or space. Any unrecognized character is copied to the target as is.
- Enclose special characters in single or double quotes, because the resulting value is passed to the target as a character string.
- Allow qualifiers to contain a *null* terminated string. The string is limited only by the buffer size (1k).

The following table shows qualifier definitions.

Table 10-5: Special date and time qualifiers

Qualifier	Definition
yy	Last two digits of year.
yyyy	All four digits of year.
mm	Month or minute (recognized by context).
mon	Three-character month abbreviation: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.
dd	Day.
hh	Hour.
ss	Seconds.
nnn	Milliseconds.
nnnnnn or uuuuuu	Microseconds.
am or pm	Indicates that you want an AM or PM designator included. The actual designator is inserted appropriate to the value.

Destination-template processing

The following sections describe destination-template processing using transfer from and transfer to statements.

transfer from statements

The following steps describe how a service library processes a destination-template transfer from statement from Adaptive Server to ODBC.

1. The access service executes the sourceselectstatement against the primary database and retrieves the schema of the result set.
2. The access service inserts the first *n* rows of data (where *n* is the setting of the TransferBatch property) resulting from the sourceselectstatement into the destination-template. The access service formats the data according to the specified qualifiers and groups the statements into a request.
3. The access service executes the request against the primary database.
4. The access service substitutes the next n rows and executes another request. It repeats this process until all the rows finish processing.

5 If conversion errors occur as rows are inserted, the invalid rows are handled according to the values set in the CharConvertError, NumConvertError, DatetimeConvertError, DefaultDate, DefaultNum, and DefaultTime properties, and the transfer continues. If SendWarningMessages is set to yes, a warning message is sent to the client.

transfer to statements

The following steps describe how a service library processes a destination-template transfer to statement to a target from ODBC.

1 The access service issues the sourceselectstatement against the primary database and receives the schema of the result set.

2 The access service receives the results of the sourceselectstatement and converts them to the predefined Adaptive Server datatypes. These datatypes do not necessarily match the datatypes of the columns in the destination table.

3 The access service substitutes the first *n* rows of the result set (where *n* is the number of rows specified in the TransferBatch configuration property). The access service formats the data according to the specified datatype qualifiers and batches the statements into a request.

4 The access service issues the request against the secondary database.

5 The access service substitutes the next n rows into the *destinationtemplatestatement* and issues another request against the secondary database. It repeats this process until all the rows finish processing.

6 If conversion errors occur as rows are inserted, the invalid rows are handled according to the values set in the CharConvertError, NumConvertError, DatetimeConvertError, DefaultDate, DefaultNum, and DefaultTime properties, and the transfer continues. If SendWarningMessages is set to yes, a warning message is sent to the client.

Datatype conversion for *transfer to* statements

Datatype conversion takes place in two steps. The following table shows how incoming target ODBC datatypes are initially converted to Open Server datatypes.

Table 10-6: Datatype conversion for transfer to statements

ODBC datatype	Open Server datatype
CHAR	CS_CHAR
VARCHAR	CS_CHAR
LONGVARCHAR	CS_TEXT
SMALLINT	CS_SMALLINT
INT	CS_INT
DECIMAL	CS_DECIMAL
DOUBLE	CS_FLOAT
REAL	CS_REAL
FLOAT	CS_FLOAT
DATE	CS_CHAR
TIME	CS_CHAR
TIMESTAMP	CS_CHAR
BINARY	CS_BINARY
VARBINARY	CS_VARBINARY
LONGVARBINARY	CS_IMAGE
TINYINT	CS_TINYINT
BIT	CS_BIT
BIGINT	CS_FLOAT
NUMERIC	CS_NUMERIC

These datatypes are converted into the datatypes specified by the destinationtemplatestatement datatype qualifiers.

Destination-template transfer errors

Value errors occur during transfer processing when the value being inserted is out of range for the column’s datatype. The access service handles these errors using the following properties:

- CharConvertError

- NumConvertError
- DateTimeConvertError
- DefaultDate
- DefaultTime
- DefaultNum

If the SendWarningMessages property is set to yes, the access service sends a message to the client application when it encounters value errors. In addition, these properties can be used to fill in default values when datatype conversion fails during both types of transfer.

Obtaining error information

You can obtain destination-template transfer error information in the following ways:

- If you include with report in the transfer statement, you receive a result set containing one VARCHAR column and one row indicating the number of rows transferred, rejected, and modified during processing.
- You can execute @@RejectedRowCount or @@DefaultedRowCount immediately after a successful transfer. These global variables return the number of rejected or defaulted rows.
- If you set SendWarningMessages to yes, the access service returns data conversion errors to the client application.

During processing, the access service sets the StopCondition property to none. It uses the value in the TransferErrorCount property to determine the number of error rows it will allow before it stops processing.

You can set this value with the following statement:

```
set TransferErrorCount nnn
```

The default setting is 0, which causes the access service to ignore errors.

Creating a transfer RPC

The client application can create a stored procedure within Adaptive Server that invokes the DirectConnect access service library transfer function. The access service library receives the transfer command as an RPC event with the following arguments:

- Name of the RPC: “transfer”
- Argument 1: The secondary connection information ({to | from} “server userid pw”)
- Argument 2: Either the bulk copy target command or the destination-template sourceselectstatement.
- Argument 3: Either the bulk copy sourceselectstatement or the destination-template sourceselectstatement.

Transfer RPC example

The following example outlines how a stored procedure shows a bulk copy transfer statement to be received as an RPC.

```
create procedure replauth as
begin
execute servername. . .transfer
“to ‘servername2 userid password’;”,
 “with replace into authors;”,
 “select * from authors;”
end
```

where:

- *servername* specifies the access service to handle the transfer. In addition:
  - The double quotes (“...”) following the *servername* are required.
  - The access service library recognizes anything other than “transfer” in the next position as the name of an ODBC stored procedure.
- *servername2* specifies the secondary database for the transfer command.

Executing a transfer RPC

A client can log in to the Adaptive Server on which this procedure is defined and invoke it as follows:

```
    execute replauth
```

When Adaptive Server executes replauth, it passes an RPC to the access service. The access service returns any result rows or messages to the client application, not to Adaptive Server.

CHAPTER 11

Accessing Catalog Information with CSPs

This chapter describes how to use catalog stored procedures (CSPs). It contains the following topics:

*Not supported in DirectConnect for DB2 UDB.

Description of CSPs

CSPs are specially recognized commands that return catalog information. Client applications use CSPs instead of SQL to access information contained in the system catalog of the target database. The access service library implements CSPs by executing stored procedures against the target catalog.

When you invoke a CSP, the access service executes a stored procedure that returns a result set. It attempts to match the results an Adaptive Server would return under the same circumstances. Because the ODBC catalog is significantly different from the Adaptive Server catalog, an exact match is impossible. To make the results match, the access service performs additional computations as the rows are returned.

Because some information is static, sp_datatype_info uses memory tables to improve the speed of operation.

Syntax

The following syntactical rules apply to CSPs:

1. Arguments can be delimited with commas and identified by position, as shown:

   ```
   sp_columns parm1,parm2
   ```

2. Arguments can be identified using the keyword NULL, as shown:

   ```
   sp_columns NULL, NULL, parm3
   ```

3. Empty arguments can be identified using empty strings, as shown:

   ```
   sp_columns ' ',' ',smith
   ```

4. Arguments can be named using the syntax @name=parm, as shown:

   ```
   sp_columns @table_owner=smith
   ```

The positional forms (1, 2, and 3) cannot be mixed with the named form (4).

The Access Service Library does not support the TABLE_QUALIFIER or PROCEDURE_QUALIFIER parameters. For all CSPs, leave the parameter empty or set it to NULL.

The argument syntax for most of the CSPs referenced in this chapter is contained in the *Sybase Adaptive Server Reference Manual, v 2*.

RPC events

You can execute CSPs using a language command or an RPC event.

When the access service processes a CSP as an RPC event, it retrieves the name and parameters from the client application using standard RPC processing techniques.

Treatment of special characters

The access service supports only the “%” wildcard character, which can be used in parameters that allow wildcard-character search patterns. The character represents any string of zero or more characters.

The access service treats all underscore characters as literals.

ODBC information

The ODBC definition of the CSPs is the model for behavior. If the access service cannot support a particular procedure, it returns the expected column form descriptors, with no data rows.

The format and content of results returned by most CSPs are described in the *MicrosoftODBC 3.5 Programmer’s Reference and SDK Guide*.

ODBC conformance levels

The three conformance levels for ODBC drivers are core, level one, and level two. CSPs that support the functionality in levels one and two are as follows:

Level one

- sp_columns
- sp_datatype_info
- sp_special_columns
- sp_statistics
- sp_tables

Level two

- sp_column_privileges
- sp_fkeys
- sp_pkeys
- sp_sproc_columns

- sp_stored_procedures
- sp_table_privileges

Compatibility

To maintain compatibility with Adaptive Server and previous versions of the MDI Database Gateway, all CSPs accomplish the same tasks as their counterparts in the other systems. Areas in which DirectConnect differs are as follows:

- The result sets returned conform to ODBC requirements.
- The result sets returned conform to ASE/CIS requirements.

When a CSP requirements conflict arises, the following precedence rules are used, in order of importance:

1. Make the results conform to ODBC specifications.
2. Make the results conform to ASE/CIS specifications.
3. Make the behavior of the procedure conform to its counterpart in Adaptive Server.

sp_column_privileges

Description

Returns column privilege information for one or more columns in a table or view.

Note This stored procedure is not supported in either DirectConnect for DB2 UDB.

Syntax

sp_column_privileges *table_name* [, *table_owner*
[, *table_qualifier* [, *column_name*]]]

Parameters

table_name
is the name of the table. Wildcard-character search patterns are not supported.

table_owner
is the name of the table owner. Wildcard-character search patterns are not supported.

table_qualifier
is ignored. Leave blank or set to NULL.

column_name
is the name of the column for which you want privilege information. Use wildcard-character search patterns to request information about more than one column. Leave blank or set to NULL to request information about all columns in the table or tables.

Usage

This procedure corresponds to the ODBC function called SQLColumnPrivileges.

sp_columns

Description

Returns information about the type of data that can be stored in one or more columns.

Syntax

sp_columns *table_name* [, *table_owner*]
 [, *table_qualifier*] [, *column_name*]

Parameters

table_name
is the table name or view. Use the wildcard character to request information about more than one table.

table_owner
is the owner of the table or view. Use the wildcard character to request information about tables owned by more than one user.

table_qualifier
is ignored. Leave blank or set to NULL.

column_name
is the name of the column for which you want information. Use the wildcard character to request information about more than one column.

Usage

- This procedure returns the Adaptive Server datatype that most clearly matches the native datatype of the target, regardless of the current datatype properties.
- This procedure corresponds to the ODBC function SQLColumns.

Results

This procedure returns one row containing a description of each column in a table.

- There are three columns in the result set that describe each columns data type; TYPE_NAME, DATA_TYPE, and REMOTE_DATA_TYPE.
- Table 11-1 describes the values returned in the TYPE_NAME and DATA_TYPE columns of the result set. TYPE_NAME contains the ODBC data type name and DATA_TYPE contains the ODBC integer identifier.
- The REMOTE_DATA_TYPE column contains a 32-bit or 4-byte composite user datatype UDT specifically identifying the remote datatype. See Table 11-2.

Note Table 11-1 also describes the identifiers returned in the TYPE_NAME and DATA_TYPE columns in the result set for sp_special_columns.

Table 11-1: ODBC Datatypes

ODBC datatype (TYPE_NAME)	Target datatype length	DATA-TYPE	ODBC type	Sybase type
BINARY	254	(-2)	SQL_BINARY	CS_BINARY
VARBINARY	254	(-3)	SQL_VARBINARY	CS_VARBINARY
LONGVARBINARY	2^31	(-4)	SQL_LONGVARBINARY	CS_LONGBINARY
CHAR()	254	(1)	SQL_CHAR	CS_CHAR
VARCHAR	254	(12)	SQL_VARCHAR	CS_VARCHAR
LONGVARCHAR	2^31	(-1)	SQL_LONGVARCHAR	CS_LONGCHAR
SMALLINT	2	(5)	SQL_SMALLINT	CS_SMALLINT
INTEGER	4	(4)	SQL_INTEGER	CS_INT
DOUBLE	8	(8)	SQL_DOUBLE	CS_FLOAT
FLOAT()	8	(6)	SQL_FLOAT	CS_FLOAT
REAL	4	(7)	SQL_REL	CS_REAL
DECIMAL()	17	(3)	SQL_DECIMAL	CS_DECIMAL
NUMERIC	17	(2)	SQL_NUMERIC	CS_NUMERIC
DATE	4	(9)	SQL_DATE	CS_DATE
TIME		(10)	SQL_TIME	CS_TIME
TIMESTAMP	10	(11)	SQL_TIMESTAMP	CS_DATETIME
TINYINT	1	(-6)	SQL_TINYINT	CS_TINYINT
BIGINT	19	(-5)	SQL_BIGINT	CS_FLOAT
BIT	1	(-7)	SQL_BIT	CS_BIT

ODBC datatype (TYPE_NAME)	Target datatype length	DATA-TYPE	ODBC type	Sybase type
GUID	36	-11	SQL_GUID	CS_CHAR
WCHAR	254	-8	SQL _WCHAR	CS_UNICHAR
WVARCHAR	254	-9	SQL _WVARCHAR	CS_UNICHAR
WLONGVARCHAR	230	-10	SQL _WLONGVARCHAR	CS_UNITEXT

This procedure allows transmission of column datatypes using a target-specific-type ID. The REMOTE_DATA_TYPE column contains a 32-bit composite datatype defined by the access service.

The following table describes the datatype value.

Table 11-2: REMOTE_DATA_TYPE return value

Bits	Value returned
Bits 0-7	ODBC datatype (can be extended for types not defined in ODBC).
Bit 8	1 if nullable, 0 if not nullable.
Bit 9	1 if case sensitive, 0 if not case sensitive.
Bits 10, 11	10 (binary); ability to be updated unknown.
Bits 12, 13	Reserved; always returns 00 (binary). The access service bulk copy feature uses this.
Bits 14, 15	01 (binary); NEWODBCDATATYPE (used for all except REAL) 10 (binary); NEWUSERTYPE (used for REAL).
Numeric types: Bits 17–23 Bits 24–31	 Precision. Scale.
Non-numeric types: Bits 16–31	Length.

sp_databases

Description Returns a list of databases on a target DBMS.

Syntax sp_databases

Parameters This procedure does not allow parameters.

Usage

Not defined.

sp_datatype_info

Description

Returns information about a particular datatype or all supported datatypes.

Syntax

sp_datatype_info [*data_type*]

Parameters

data_type
is the ODBC code number for the specified datatype for which sp_datatype_info returns information. See Table 11-1 for a description of these codes.

Usage

- The *data_type* parameter specifies the ODBC datatype for which information is requested. If this parameter is not provided, sp_datatype_info returns information about all supported datatypes.
- This procedure corresponds to the ODBC function SQLGetTypeInfo.
- The DatatypeInfo property specifies whether information is returned for Transact SQL datatypes or target database datatypes. For configuration information, see "DatatypeInfo" on page 24.

  If the value for *data_type* equals:

  - target, the sp_datatype_info returns all target datatypes and their associated ODBC datatypes. A specific ODBC datatype may be used to represent multiple target datatypes.
  - transact, the sp_datatype_info returns the T-SQL datatype that best matches each ODBC datatype that the target represents.

Results sp_datatype_info returns a list of datatypes with information about each. Results are ordered by the following columns:

- DATA_TYPE
- TYPE_NAME

The lengths for varchar columns shown in the result set tables are maximums; the actual lengths depend on the target database.

The following table shows the result set.

Table 11-3: Result set for sp_datatype_info

Column	Datatype	Description
TYPE_NAME	varchar(128) NOT NULL	Name of the T-SQL datatype or the target database datatype that corresponds to the ODBC datatype in the DATA_TYPE column.
DATA_TYPE	smallint NOT NULL	ODBC datatype to which all columns of this type are mapped.
PRECISION	int	Maximum precision allowed for this datatype; NULL is returned for datatypes where precision is not applicable.
LITERAL_PREFIX	varchar(128)	Character(s) used to prefix a literal; NULL is returned for datatypes where a literal prefix is not applicable.
LITERAL_SUFFIX	varchar(128)	Character(s) used to mark the end of a literal; NULL is returned for datatypes where a literal suffix is not applicable.
CREATE_PARAMS	varchar(128)	Description of the creation parameters required for this datatype (for example: precision and scale); NULL is returned if the datatype does not have creation parameters.
NULLABLE	smallint NOT NULL	Indicates whether the datatype accepts NULL values: • 0 means the column does not accept NULL values. • 1 means the column accepts NULL values.
CASE_SENSITIVE	smallint NOT NULL	Indicates whether the datatype distinguishes between uppercase and lowercase characters: • 0 means the datatype is not a character type or is not case sensitive. • 1 means the datatype is a character type and is case sensitive.
SEARCHABLE	smallint NOT NULL	Indicates how this datatype is used in where clauses: • 0 means the datatype cannot be used in a where clause. • 1 means the datatype can be used in a where clause.
UNSIGNED_ATTRIBUTE	smallint	Indicates whether this attribute is unsigned: • 0 means the datatype is signed. • 1 means the datatype is unsigned. • NULL means the datatype is not numeric.

Column	Datatype	Description
MONEY	smallint NOT NULL	Indicates whether this is a money datatype: • 0 means it is not a money datatype. • 1 means it is a money datatype.
AUTO_INCREMENT	smallint	Indicates whether this datatype automatically increments: • 0 means columns of this datatype do not automatically increment. • 1 means columns of this datatype automatically increment. • NULL means the column is not numeric and does not have a sign.
LOCAL_TYPE_NAME	varchar(128)	The database name or the T-SQL name for the datatype.
MINIMUM_SCALE	smallint	Minimum scale for the datatype; NULL if scale is not applicable.
MAXIMUM_SCALE	smallint	Maximum scale for the datatype; NULL if scale is not applicable.

sp_fkeys

Description

Returns primary and foreign key information for the specified table or tables. Foreign keys must be declared through the ANSI integrity constraint mechanism.

Syntax

sp_fkeys *pktable_name* [, *pktable_owner*]
 [, *pktable_qualifier*] [, *fktable_name*]
 [, *fktable_owner*] [, *fktable_qualifier*]

Parameters

pktable_name
is the name of the table containing the primary key. Wildcard-character search patterns are not supported. You must specify this parameter, the fktable_name parameter, or both.

pktable_owner
is the owner of the table containing the primary key. Wildcard-character search patterns are not supported.

pktable_qualifier
is ignored. Leave blank or set to NULL.

fktable_name
is the name of the table containing the foreign key. Wildcard-character search patterns are not supported. You must specify this parameter, the pktable_name parameter, or both.

fktable_owner
is the owner of the table containing the foreign key. Wildcard-character search patterns are not supported.

fktable_qualifier
is ignored. Leave blank or set to NULL.

Usage
This procedure corresponds to the ODBC function SQLForeignKeys.

sp_pkeys

Description
Returns primary key information for a single table. Primary keys must be declared through the ANSI integrity constraint mechanism.

Syntax
sp_pkeys *table_name* [, *table_owner*]
[, *table_qualifier*]

Parameters
table_name
is the name of the table. Wildcard-character search patterns are not supported.

table_owner
is the owner of the table. Wildcard-character search patterns are not supported.

table_qualifier
is ignored. Leave blank or set to NULL.

Usage
This procedure corresponds to the ODBC function SQLPrimaryKeys.

sp_server_info

Description
Returns target server metadata containing a list of attribute names and matching values for the target.

Syntax
sp_server_info [*attribute_id*]

Parameters

attribute_id
is the integer ID of the attribute.

Usage

This procedure generates an extensible result set. It can be expanded, depending upon the needs of the specific access service library.

sp_special_columns

Description

Retrieves the following information about columns within a specified table or view:

- The optimal set of columns that uniquely identifies a row in the table or view
- The columns that are automatically updated when any value in the row is updated by a transaction

Syntax

sp_special_columns *table_name* [, *table_owner*]
 [, *table_qualifier*] [, *col_type*]

Parameters

table_name
is the name of the table. Wildcard-character search patterns are not supported.

table_owner
is the owner of the table. Wildcard-character search patterns are not supported.

table_qualifier
is ignored. Leave blank or set to NULL.

col_type
is a value that requests information about columns of a specific type as follows:

- "R" returns information about columns with values that uniquely identify any row in the table.
- "V" returns information about columns with values that are automatically generated by a target each time a row is inserted or updated.

Usage

- This procedure corresponds to the ODBC function SQLSpecialColumns.

- See Table 11-1 for ODBC datatypes and matching ODBC integer identifiers returned in the TYPE_NAME and DATA_TYPE columns of the result set.

sp_sproc_columns

Description

Returns information about stored procedure input and return parameters.

Syntax

Note DirectConnect supports syntax for SQL Server versions 10 and 11.

sp_sproc_columns *sp_name* [, *sp_owner*]
 [, *sp_qualifier*] [, *column_name*]

or

sp_sproc_columns *procedure_name* [, *procedure_owner*]
 [, *procedure_qualifier*] [, *column_name*]

Parameters

sp_name or *procedure_name*
is the name of the stored procedure. Wildcard-character search patterns are not supported.

sp_owner or *procedure_owner*
is the owner of the stored procedure. Wildcard-character search patterns are not supported.

sp_qualifier or *procedure_qualifier*
is ignored. Leave blank or set to NULL.

column_name
is the name of the parameter about which you want information. If you do not supply a parameter name, this procedure returns information about all input parameters.

Usage

This procedure corresponds to the ODBC function SQLProcedureColumns.

Note In DirectConnect, sp_sproc_columns returns extra, unsolicited columns.

sp_statistics

Description	Returns a list of indexes in a single table.
Syntax	sp_statistics *table_name* [, *table_owner*] [, *table_qualifier*] [, *index_name*] [, *is_unique*]

Parameters

table_name
is name of the table. Wildcard-character search patterns are not supported.

table_owner
is the owner of the table.

table_qualifier
is ignored. Leave blank or set to NULL.

is_unique
is one of the following values:

- "Y" if unique indexes are to be returned.
- "N" if unique indexes are not to be returned.

index_name
is the name of the index. Wildcard-character search patterns are not supported.

Usage

Note With all platforms in DirectConnect, the *index_name* parameter is ignored, regardless of the value. This applies even if you set the value to a nonexistent name or to NULL.

This procedure corresponds to the ODBC function SQLStatistics.

sp_stored_procedures

Description	Returns a list of available procedures.
Syntax	sp_stored_procedures [*sp_name*] [, *sp_owner*] [, *sp_qualifier*]

Parameters

sp_name
is the stored procedure name. Use the wildcard character to request information about more than one stored procedure.

sp_owner
is the owner of the stored procedure. Use the wildcard character to request information about procedures owned by more than one user.

sp_qualifier
is the name of the database. Acceptable values are the current database or NULL.

Usage

This procedure corresponds to the ODBC function SQLProcedures.

sp_table_privileges

Description

Returns privilege information for all columns in a table or view.

Note This stored procedure is not supported in either DirectConnect for DB2 UDB.

Syntax

sp_table_privileges *table_name* [, *table_owner*
[, *table_qualifier*]]

Parameters

table_name
is the name of the table. Wildcard-character search patterns are not supported.

table_owner
is the name of the table owner. Wildcard-character search patterns are not supported.

table_qualifier
For DB2 UDB targets – is ignored. Leave blank or set to NULL.

For non-DB2 targets – is the name of the database. Acceptable values are the current database or NULL.

Usage

This procedure corresponds to the ODBC function SQLTablePrivileges.

sp_tables

Description

Returns a list of objects that can appear in a from clause.

Syntax

sp_tables [*table_name*] [, *table_owner*]
[, *table_qualifier*] [, *table_type*]

Parameters

table_name
is the name of the table. Use the wildcard character to request information about more than one table.

table_owner
is the owner of the table. Use the wildcard character to request information about tables owned by more than one user.

table_qualifier
For DB2 UDB targets – is ignored. Leave blank or set to NULL.

For non-DB2 targets – is the name of the database. Acceptable values are the current database or NULL.

table_type
is a list of values, separated by commas, that gives information about all tables of the types specified, including the following:

"'TABLE', 'SYSTEM TABLE', 'VIEW', 'SYNONYM'"

Usage

- Enclose each table type with single quotation marks and enclose the entire parameter with double quotation marks. Enter table types in uppercase.
- This procedure corresponds to the ODBC function SQLTables.

CHAPTER 12 Retrieving Information with System Procedures

This chapter describes how to use system procedures to retrieve information. It contains the following topics:

Description of system procedures

System procedures are Sybase-supplied stored procedures that return information about the access service and the target database. If the access service cannot support one of these procedures, the procedure returns a formatted result set containing zero rows.

ODBC and ASE/CIS use system procedures to obtain information about DirectConnect capabilities.

System procedures in DirectConnect and Adaptive Server

System procedures for DirectConnect and their relationship with Adaptive Server and ASE/CIS are defined in the following sections. A number of system procedures are either supported, not-supported, or defined differently between DirectConnect and Adaptive Server:

Supported in DirectConnect and not in Adaptive Server

Following is a system procedure that is supported in DirectConnect, but not supported in Adaptive Server:

- sp_groups

Defined differently in DirectConnect and Adaptive Server

Following are system procedures defined differently in DirectConnect and Adaptive Server:

- sp_configure (read only in DirectConnect)
- sp_helpserver
- sp_sqlgetingo

Defined in DirectConnect for ASE/CIS

Following are system procedures defined in DirectConnect to support ASE/CIS products:

- sp_capabilities
- sp_thread_props

Not supported in DirectConnect

Following are system procedures that are not supported in DirectConnect:

- sp_char_length
- sp_datalength
- sp_password
- sp_patindex
- sp_textvalid
- sp_who

sp_capabilities

Description

Returns the SQL capabilities of an access service.

Syntax

sp_capabilities

Parameters

None.

This procedure does not allow parameters.

Usage

- The result set contains information that allows applications to successfully interact with an access service during normal query processing.
- For information about the requirements for sp_capabilities, see the *Component Integration Services User's Guide for ASE and OmniConnect*.

Results The following table shows the result set:

Table 12-1: Result set for sp_capabilities

Column	Datatype	Description
ID	int	Capability ID
CAPABILITY_NAME	char(30)	Capability name
VALUE	int	Capability value
DESCRIPTION	char(128)	Capability description

The following table shows values for the capabilities:

Table 12-2: Values for sp_capabilities

Capability number	Capability	Description
101	SQL syntax	• 1 = Sybase supported • 2 = DB2 UDB supported
102	join handling	• 0 = unsupported • 1 = all but outer join supported • 2 = full join supported
103	aggregate handling	• 0 = unsupported • 1 = count not supported • 2 = all functions
104	AND predicates	• 0 = unsupported • 1 = supported
105	OR predicates	• 0 = unsupported • 1 = supported
106	LIKE predicates	• 0 = unsupported • 1 = ANSI supported • 2 = Sybase supported
107	bulk insert handling	• 0 = unsupported • 1 = supported

Capability number	Capability	Description
108	text/image handling	• 0 = unsupported • 1 = text without textptr supported • 2 = text with textptr supported
109	transaction handling	• 0 = unsupported • 1 = local supported • 2 = two-phase-commit supported
110	text pattern handling	• 0 = unsupported • 1 =supported
111	order by	• 0 = unsupported • 1 = supported
112	group by	• 0 = unsupported • 1 = supported
113	net password encryption	• 0 = unsupported • 1 = supported
114	object case sensitivity	• 0 = case-insensitive • 1 = case-sensitive
115	distinct	• 0 = unsupported • 1 = supported
116	wildcard escape	• 0 = unsupported • Anything else is the escape character
117	union handling	• 0 = unsupported • 1 = supported
118	string functions	• 0 = unsupported • 1 = substring supported • 2 = Oracle subset supported • 3 = all TransacTransact-SQL supported
119	expression handling	• 0 = unsupported • 1 = supported • 2 = full Transact-SQL supported
120	truncate trailing spaces on varchar parameters	• 0 =no • 1 = yes
121	language events	• 0 = no support for DML • 1 = DML support without datetime in where clause • 2 = no restrictions on DML

Capability number	Capability	Description
122	date functions	• 0 = unsupported • 1 = all Transact-SQL supported
123	math functions	• 0 = unsupported • 1 = Oracle subset supported • 2 = all Transact-SQL supported
124	Transact-SQL convert function	• 0 = unsupported • 1 = supported
125	Transact-SQL delete/update	• 0 = unsupported • 1 = Transact-SQL extensions supported
126	insert/update handling	• 0 = unsupported • 1 = supported
127	subquery handling	• 0 = unsupported • 1 = supported
128	in/not in clause	• 0 = unsupported • 1 = supported
129	case expression in a SQL statement	• 0 = unsupported • 1 = supported
132	tables per statement	0=Unsupported 1=Supported
133	Java UDF support	0=Unsupported 1=Supported
134	Java ADT support	0=Unsupported 1=Supported
135	Quoted Identifier support	0=Unsupported 1=Supported
137	SELECT-NULL support	0=Unsupported 1=Supported
138	Identity column support	0=Unsupported 1=Supported

sp_configure

Description

Provides a complete list of configuration names, minimum and maximum values, configured values, and current run values for each item.

Syntax sp_configure

Parameters None.

This procedure does not allow parameters.

Usage This procedure returns an empty result set since none of the configuration information is supported.

sp_groups

Description Returns the current user name as the sole user group.

Syntax sp_groups

Parameters None.

This procedure does not allow parameters.

Usage This procedure was created for DirectConnect. It is not documented in any Adaptive Server or ODBC manuals.

Results The following table shows the result set:

Table 12-3: Result set for sp_groups

Column	Datatype	Description
GROUP_NAME	char(8)	Group Name (Authorization ID)

sp_helpserver

Description Returns the following information:

- The name of the access service in use
- The version of Open Server in use
- The version of the DirectConnect server in use
- The version of the DirectConnect Access Service Library in use
- The version of the DBMS with which the access service is associated

Syntax sp_helpserver

Parameters None.

This procedure does not allow parameters.

Usage

Not defined.

sp_sqlgetinfo

Description

Provides information about SQL grammar, syntax, and capabilities that the target DBMS supports.

Syntax

sp_sqlgetinfo [*attribute_name*]

Parameters

attribute_name
is the name of a particular SQL option.

Usage

- This function corresponds to the ODBC function SQLGetInfo.
- If this procedure is called but no option is specified, the result set includes all SQL options.
- If the attribute is not found in the internal table, the access service returns an error.
- If the parameter is not provided, the access service returns a result set of all supported SQL options.

Results Result set information is described in the following sections.

Format The format is shown in the following table.

Table 12-4: Format for sp_sqlgetinfo

sql_option	varchar(30)	not null
sql_value	varchar(255)	null

If the sql_value column is NULL, this option is not supported for the target DBMS.

SQL options are shown in the following two tables. The first table lists DirectConnect options and SQL options A through L.

Table 12-5: SQL options for sp_sqlgetinfo (A through L)

SQL option	Description
ICD_Cursor_Support	Bitmask indicating cursor support.
ICD_Dynamic_Support	Bitmask indicating dynamic statement support.
ICD_Execdirect	Bitmask indicating how dynamic execdirect statement is supported.
ICD_Language_Support	Bitmask indicating language statement support. No parameter marker support.
ICD_Longtypes_Supported	Support for long types as parameters.

SQL option	Description
ICD_Modify_Groupby	Intersolv driver insures GROUP BY clause when aggregate functions are used as part of the select list.
SQL_Accessible_Procedures	User can execute all procedures returned by sp_stored_procedures.
SQL_Accessible_Tables	User is guaranteed SELECT privileges to tables returned by sp_tables.
SQL_Active_Connections	No known limit to the number of connections.
SQL_Active_Statements	No known limit to the number of statements for a connection.
SQL_Alter_Table	Bitmask indicating which clauses in ALTER TABLE are supported.
SQL_Bookmark_Persistence	Bitmask enumerating through which bookmarks persist. None supported.
SQL_Column_Alias	Support for column alias.
SQL_Concat_Null_Behavior	Bitmask indicating how the DBMS handles concatenations with NULLS.
SQL_Convert_Bigint SQL_Convert_Binary SQL_Convert_Bit SQL_Convert_Char SQL_Convert_Date SQL_Convert_Decimal SQL_Convert_Double SQL_Convert_Float SQL_Convert_Integer SQL_Convert_Longvarbinary SQL_Convert_Longvarchar SQL_Convert_Numeric SQL_Convert_Real SQL_Convert_Smallint SQL_Convert_Time SQL_Convert_Timestamp SQL_Convert_Tinyint SQL_Convert_Varbinary SQL_Convert_Varchar	Bitmask indicating conversions “to type” supported.
SQL_Convert_Functions	Bitmask indicating conversion functions supported.
SQL_Correlation_Name	Table correlation names supported.
SQL_CSP_Support	Sybase/Intersolv extension for supporting CSPs. Value = 16383.
SQL_Cursor_Commit_Behavior	Bitmask indicating how a commit operation affects a cursor.
SQL_Cursor_Rollback_Behavior	Bitmask indicating how a rollback operation affects a cursor.
SQL_Cursor_Sensitivity	A value indicating support for cursor sensitivity.
SQL_Database_Name	Value provided by the DirectConnect server.
SQL_Date_Source_Read_Only	The data source is read/write.
SQL_DBMS_Name	The target DBMS name. A maximum of 30 characters is returned.

SQL option	Description
SQL_DBMS_Ver	The target DBMS version in the form ##.##.####. A maximum of 30 characters is returned. The version string may have target-specific information that follows.
SQL_Default_TXN_Isolation	Bitmask indicating the default transaction level supported by the DBMS.
SQL_Dynamic_Cursor_Attributes1	Bitmask that describes the attributes of a dynamic cursor that are supported by the driver (1st subset of attributes.)
SQL_Dynamic_Cursor_Attributes2	Bitmask that diatribes the attributes of a dynamic cursor that are supported by the driver (2nd subset of attributes.)
SQL_Expressions_In_Orderby	Support for expressions in order by clause.
SQL_Fetch_Direction	Bitmask enumerating supported options.
SQL_File_Usage	Files treated in data source.
SQL_Forward_Only_Cursor_Attributes1	A bitmask that describes the attributes of a forward-only cursor that are supported by the driver (1st subset of attributes.)
SQL_Forward_Only_Cursor_Attributes2	A bitmask that describes the attributes of a forward-only cursor that are supported by the driver (2nd subset of attributes.)
SQL_Getdata_Extensions	Bitmask enumerating extensions to SQLGetData.
SQL_Group_By	Bitmask indicating the relationship between GROUP BY columns supported in the DBMS.
SQL_Identifier_Case	Defines whether identifiers are case sensitive.
SQL_Identifier_Quote_Char	Character used to delimit quoted identifiers.
SQL_Keywords	See the *Microsoft ODBC 3.5 Programmer's Reference and SDK Guide* for information.
SQL_Like_Escape_Clause	Support of "%" character and "_" character as escape characters in like clause.
SQL_Lock_Types	Bitmask enumerating supported lock types.

The second table lists SQL options M through Z for sp_sqlgetinfo.

Table 12-6: SQL options for sp_sqlgetinfo (M through Z)

SQL option	Description
SQL_Max_Binary_Literal_Len	Maximum length of binary literal is either unknown or unlimited.
SQL_Max_Char_Literal_Len	Maximum length of character literal is either unknown or unlimited.
SQL_Max_Column_Name_Len	Maximum length for a column name. Convert this string to an integer. A value of 0 means not supported.
SQL_Max_Columns_In_Group_By	Maximum number of columns allowed in a SQL GROUP BY clause. Convert this string to an integer. A value of 0 means that the limit is unknown or unlimited.

SQL option	**Description**
SQL_Max_Columns_In_Index	Maximum number of columns allowed in a SQL CREATE INDEX. Convert this string to an integer. A value of 0 means that the limit is unknown.
SQL_Max_Columns_In_Order_By	Maximum number of columns allowed in a SQL ORDER BY clause. Convert this string to an integer. A value of 0 means that the limit is unknown.
SQL_Max_Columns_In_Select	Maximum number of columns allowed in a SQL SELECT column list. Convert this string to an integer. A value of 0 means that the limit is unknown.
SQL_Max_Columns_In_Table	Maximum number of columns allowed in a SQL CREATE TABLEE. Convert this string to an integer. A value of 0 means that the limit is unknown.
SQL_Max_Cursor_Name_Len	Maximum length for a cursor name. Convert this string to an integer. A value of 0 means not supported.
SQL_Max_Index_Size	Maximum number of characters allowed in the combined column length of an index. Convert this string to an integer. A value of 0 indicates that the limit is unknown.
SQL_Max_Identifier_Len	Maximum size in characters that the data source supports for user-defined names.
SQL_Max_Owner_Name_Len	Maximum length for an owner name. Convert this string to an integer. A value of 0 means not supported.
SQL_Max_Procedure_Name_Len	Maximum length for a procedure name. Convert this string to an integer. A value of 0 means not supported.
SQL_Max_Qualifier_Name_Len	Maximum length for a qualifier name. Convert this string to an integer. A value of 0 means not supported.
SQL_Max_Row_Size	Maximum number of characters allowed in the combined column length of a row in a table. Convert this string to an integer. A value of 0 means that the limit is unknown.
SQL_Max_Row_Size_Includes_Long	Includes the length of all long datatypes.
SQL_Max_Statement_Len	Maximum length allowed for a SQL statement. Convert this string to an integer. A value of 0 means that the limit is unknown.
SQL_Max_Table_Name_Len	Maximum length allowed for a table name. Convert this string to an integer. A value of 0 means not supported.
SQL_Max_Tables_In_Select	Maximum number of columns allowed in a SQL SELECT FROM clause. Convert this string to an integer. A value of 0 means that the limit is unknown.
SQL_Max_User_Name_Len	Maximum length for the user name. Convert this string to an integer. A value of 0 means not supported.
SQL_Mult_Result_Sets	Driver does not support multiple result sets in a given language event.
SQL_Multiple_Active_TXN	Only one connection can have an active transaction.
SQL_Need_Long_Data_Len	Need the length of the long datatypes.

SQL option	Description
SQL_Non_Nullable_Columns	Bitmask indicating whether the DBMS supports non-nullable columns.
SQL_Null_Collation	Bitmask indicating how the DBMS collates NULL values.
SQL_Numeric_Functions	Bitmask indicating the supported scalar numeric functions.
SQL_ODBC_API_Conformance	Bitmask enumerating ODBC level.
SQL_ODBC_SAG_CLI_Conformance	Bitmask enumerating compliance to functions of the SAG specification.
SQL_ODBC_SQL_Conformance	Bitmask indicating supported SQL grammar.
SQL_ODBC_SQL_Opt_IEF	Support for Integrity Enhancement Facility (IEF).
SQL_Order_By_Columns_In_Select	Columns in ORDER BY clause must be in select list.
SQL_OJ_Capabilities	A bitmask enumerating the types of outer joins supported by the driver and data source.
SQL_Outer_Joins	Support for outer joins.
SQL_Owner_Term	The DBMS term for an owner name. A maximum of 30 characters is returned. A null value means not supported.
SQL_Owner_Usage	Bitmask indicating statements in which owners can be used.
SQL_Pos_Operations	Bitmask enumerating the operations in SQLSetPos.
SQL_Positioned_Statements	Bitmask indicating supported positioned SQL statements.
SQL_Procedure_Term	DBMS term for a procedure name. A maximum of 30 characters is returned. A null value means not supported.
SQL_Procedures	Support for procedures.
SQL_Qualifier_Location	Bitmask indicating the position of the qualifier in a qualified table name.
SQL_Qualifier_Name_Separator	Character or string separator between the qualifier and the name element. A maximum of five characters is returned.
SQL_Qualifier_Term	DBMS term for a qualifier name. A maximum of 30 characters is returned.
SQL_Qualifier_Usage	Bitmask indicating in which statements a qualifier can be used.
SQL_Quoted_Identifier_Case	Bitmask describing SQL identifier case and storage in system tables when used in SQL statements.
SQL_Row_Updates	See the *Microsoft ODBC 3.5 Programmer's Reference and SDK Guide* for information.
SQL_Scroll_Concurrency	Bitmask identifying concurrency control options for scrollable cursors.
SQL_Scroll_Options	Bitmask indicating scroll options for scrollable cursors.
SQL_Search_Pattern_Escape	See the *Microsoft ODBC 3.5 Programmer's Reference and SDK Guide* for information.
SQL_Set_Database_Context	Sybase/Intersolv extension for supporting CSPs. If value = Y, the driver issues use_database_name to the configured database name and is sensitive to three-part names.
SQL_Special_Characters	Special characters used in object names. All characters except a-z, A-Z, 0-9, and the underscore character.
SQL_SQL_Conformance	A value indicating the level of SQL-92 supported by the driver.

SQL option	**Description**
SQL_String_Functions	Bitmask indicating supported scalar string functions.
SQL_Subqueries	Bitmask indicating predicates that support subqueries.
SQL_System_Functions	Bitmask indicating supported scalar system functions.
SQL_Table_Term	DBMS term for a table name. A maximum of 30 characters is returned.
SQL_TimeDate_Add_Intervals	Bitmask indicating supported timestamp intervals associated with TIMESTAMPADD function.
SQL_TimeDate_Diff_Intervals	Bitmask indicating supported timestamp intervals associated with TIMESTAMPDIFF function.
SQL_TimeDate_Functions	Bitmask indicating supported timestamp intervals.
SQL_TXN_Capable	Indicates the transaction support in the DBMS.
SQL_TXN_Isolation_Option	Bitmask indicating transaction isolation levels.
SQL_Union	Bitmask indicating support for union clause.
SQL_User_Name	Current user name. A maximum of SQL_Max_User_Name_Len characters are returned. A null value means not supported.

sp_thread_props

Description

Enables the client to retrieve and set various thread properties.

Syntax

sp_thread_props [*property_name* [, *property_value*]]

Parameters

property_name
is the name of the property to be set or shown.

property_value
is the value to which the property is to be set.

Usage

If you do not provide any parameters, or if you provide only *property_name,* the access service returns a single result set consisting of every instance of *property_name* and the value for each.

APPENDIX A

Configuration Quick Reference

This appendix contains a quick reference table of the configuration properties, in alphabetical order.

Quick Reference table

The following Quick Reference table shows all the DirectConnect configuration properties for:

- DirectConnect DB2 UDB
- DirectConnect for Informix
- DirectConnect for Microsoft SQL Server
- DirectConnect for ODBC

In the following table, the property category represents the subsection heading in the access service library configuration file for DirectConnect targets.

For detailed explanations of configuration properties, see Chapter 2, "Configuring the Access Service Library for DirectConnect."

Table A-1: Configuration properties for DirectConnect

Property name	Property values	Property category {subsection name}	Global variable (GV) or set statement (SS)
Allocate	[*connect* | *request*]	{Target Interaction}	GV and SS
BinaryResults	[*binary* | *char*]	{Datatype Conversion}	GV and SS
BulkCommitCount	*integer*	{Transfer}	none
CharConvertError	[*reject* | *truncate*]	{Data Conversion Errors}	GV and SS
ClientDecimalSeparator	*char*	{Client Interaction}	GV and SS

Property name	**Property values**	**Property category {subsection name}**	**Global variable (GV) or set statement (SS)**
ClientIdleTimeout	*integer*	{Client Interaction}	none
ConnectionSpec1	*char*	{ACS Required}	none
CSPExclusions	[*none* | *user* | *nonauth* | *nonauthpublic*]	{Catalog Stored Procedures}	GV and SS
CSPIncludeAlias	[*no* | *yes*]	{Catalog Stored Procedures}	GV and SS
CSPIncludeSynonym	[*no* | *yes*]	{Catalog Stored Procedures}	GV and SS
CSPIncludeSystem	[*no* | *yes*]	{Catalog Stored Procedures}	GV and SS
CSPIncludeTable	[*yes* | *no*]	{Catalog Stored Procedures}	GV and SS
CSPIncludeView	[*yes* | *no*]	{Catalog Stored Procedures}	GV and SS
DatatypeInfo	[*transact* | *target*]	{Catalog Stored Procedures}	GV and SS
DateResults	[*datetime* | *datetime4* | *char_iso* | *char_usa* | *char_eur* | *char_jis* | *char_odb*c]	{Datatype Conversion}	GV and SS
DateTimeConvertError	[*reject* | *null* | *default*]	{Data Conversion Errors}	GV and SS
DateTimeResults	[*datetime* | *datetime4* | *char_iso* | *char_usa* | *char_eur* | *char_jis* | *char_odbc*]	{Datatype Conversion}	GV and SS
DecimalResults	[*autoconvert* | *int* | *float* | *real* | *char* | *money* | *money4* | *bcd*]	{Datatype Conversion}	GV and SS
DefaultDate	*yyyy-mm-dd*	{Data Conversion Errors}	none
DefaultNum	*integer*	{Data Conversion Errors}	none
DefaultTime	*hh.mm.ss*	{Data Conversion Errors}	none
DelimitSqlRequests	[*no* | *yes*]	{Target Interaction}	GV and SS
EnableAtStartup	[*no* | *yes*]	{Client Interaction}	none
FloatResults	[*float* | *real* | *char*]	{Datatype Conversion}	GV and SS
Int2Results	[*smallint* | *char*]	{Datatype Conversion}	GV and SS

Property name	**Property values**	**Property category {subsection name}**	**Global variable (GV) or set statement (SS)**
Int4Results	[*int* | *char*]	{Datatype Conversion}	GV and SS
IsolationLevel	[*ur* | *cr* | *rr* | *sr* | *vr* | *no*]	{Target Interaction}	GV
LogConnectionStatistics	[*no* | *yes*]	{Logging}	none
LogReceivedSQL	[*no* | *yes*]	{Logging}	none
LogRequestStatistics	[*no* | *yes*]	{Logging}	none
LogServiceStatistics	*integer*	{Logging}	none
LogSvclibStatistics	*integer*	{Logging}	none
LogTargetActivity	[*no* | *yes*]	{Logging}	none
LogTransferStatistics	[*no* | *yes*]	{Logging}	none
LogTransformedSQL	[*no* | *yes*]	{Logging}	none
MaxResultSize	*integer*	{Client Interaction}	GV and SS
MaxRowsReturned	*integer*	{Client Interaction}	GV and SS
MaxSvcConnections	*integer*	{Client Interaction}	GV
NumConvertError	[*reject* | *null* | *default*]	{Data Conversion Errors}	GV and SS
quoted_identifier	[*on* | *off*]	{Client Interaction}	GV and SS
QuotedStringDelimiter	*char*	{Target Interaction}	GV and SS
RealResults	[*float* | *real* | *char*]	{Datatype Conversion}	GV and SS
ReturnNativeError	[*no* | *yes*]	{Target Interaction}	GV and SS
SendWarningMessages	[*no* | *yes*]	{Client Interaction}	GV and SS
ServiceDescription	*char*	{Client Interaction}	GV
SQLOdbcCursors	[*if_needed* | *odbc* | *driver* | *default*]	{Target Interaction}	SS
SQLTransformation	[*passthrough* | *sybase* | *tsql0* | *tsql1* | *tsql2*]	{Target Interaction}	GV and SS
StripBinaryZero	[*yes* | *no*]	{Client Interaction}	GV and SS
StopCondition	[*error* | *none* | *warning*]	{Target Interaction}	GV and SS
SvclibDescription	*char*	{Client Interaction}	GV
TargetDBMS	[*notused* | *UDBLAN* | *UDBOS390* | *UDBAS400*]	{Target Interaction}	GV and SS
TargetDecimalSeparator	*char* (default is a period)	{Target Interaction}	GV

Property name	**Property values**	**Property category {subsection name}**	**Global variable (GV) or set statement (SS)**
TextSize	*integer*	{Client Interaction}	GV and SS
TimeResults	[*datetime* | *datetime4* | *char_iso* | *char_usa* | *char_eur* | *char_jis* | *char_odbc*]	{Datatype Conversion}	GV and SS
TinyIntResults	[*smallint* | *tinyint*]	{Datatype Conversion}	GV and SS
TraceEvents	[*no* | *yes*]	{Tracing}	none
TraceInterface	[*no* | *yes*]	{Tracing}	none
TraceTarget	[*no* | *yes*]	{Tracing}	none
TransactionMode	[*short* | *long*]	{Client Interaction}	GV and SS
TransferBatch	*integer*	{Transfer}	GV and SS
TransferErrorAction	[*noaction* | *rollback*]	{Transfer}	GV and SS
TransferErrorCount	*integer*	{Transfer}	GV and SS
Version	*versionstring*	{Client Interaction}	GV
XNLChar	*integer*	{Datatype Conversion}	GV and SS
XNLVarChar	*integer*	{Datatype Conversion}	GV and SS

APPENDIX B Converting Datatypes

This appendix describes datatype conversions that take place between ODBC and Open Server, Informix, Microsoft SQL Server, and DB2 UDB.

This appendix contains the following topics:

Limitations

The maximum size of data through DirectConnect products is 32,767 bytes for all datatypes, including text and image. Data larger than 32,767 bytes is truncated.

Warning! DirectConnect cannot correctly represent or transport varchar values containing empty strings (zero length non-null strings). Empty string varchar values are represented as *NULL* values.

ODBC-to-Open Server datatypes

When you retrieve data from the target database, the access service converts the target data to default Open Server datatypes for delivery to the client application. The following table shows ODBC datatypes and the resulting Open Server datatypes.

Table B-1: ODBC to Open Server datatype mapping

ODBC datatype	Open Server datatype
SQL_CHAR	CS_CHAR
SQL_VARCHAR	CS_VARCHAR
SQL_LONGVARCHAR	CS_TEXT
SQL_DECIMAL	CS_DECIMAL
SQL_NUMERIC	CS_NUMERIC
SQL_SMALLINT	CS_SMALLINT
SQL_INTEGER	CS_INTEGER
SQL_REAL	CS_REAL
SQL_FLOAT	CS_FLOAT
SQL_DOUBLE	CS_FLOAT
SQL_BIT	CS_BIT
SQL_TINYINT	CS_TINYINT
SQL_BIGINT	CS_NUMERIC
SQL_BINARY	CS_BINARY
SQL_VARBINARY	CS_VARBINARY
SQL_LONGVARBINARY	CS_IMAGE
SQL_DATE	CS_CHAR
SQL_TIME	CS_CHAR
SQL_TIMESTAMP	CS_CHAR
SQL_INTERVAL	CS_CHAR

Result set data value conversion

Data values returned from the target DBMS to a client application are converted into a format that Open Client and Open Server can handle. This conversion can encounter inconsistencies, particularly in supported ranges. The access service must convert the value from the target into a matching Open Client and Open Server datatype before it sends the value back to the client.

To do this, the access service uses configuration properties. Each target datatype has a default Open Client and Open Server mapping, but these may be overridden either by the configuration property (thus affecting the entire service) or through a set statement (thus affecting only the client connection).

For more information on configuration values that affect data conversion, see "Data conversion error properties" on page 31.

Data values sent to the client application

CHAR and VARCHAR datatype values shorter than the XNLCHAR and XNLVARCHAR values are returned to the client application as CS_CHAR. Datatype values longer than the XNLCHAR and XNLVARCHAR values, are returned as CS_TEXT.

A DECIMAL datatype is returned to the client application as CS_DECIMAL. Otherwise, the configuration settings shown in the DecimalResults configuration property applies.

For clients with System 10™ and earlier versions, DECIMAL data is returned as CS_FLOAT.

Open Server-to-ODBC datatypes

An access service converts or performs SQL transformation on incoming Open Server data it receives in a client request when the data includes:

- Data values embedded as strings within the text of select, insert, delete, update, and execute language commands
- Data values as parameters of RPC, cursor, transfer, or dynamic SQL commands
- Datatype names as part of create table or alter table commands

The access service does not perform automatic incoming datatype conversions on data values embedded in strings or received as parameters. Instead, the client application receives a string template from the target datatypes so that it can format the strings correctly before sending them to the target DBMS. The formatting is set up through the sp_columns catalog stored procedure.

Data values embedded as strings

When the access service receives a SQL command with embedded data values, the SQL transformation mode in effect determines whether any transformation is applied to these values. The following rules apply:

- If the access service is in passthrough mode, it does not perform transformation.

- If the access service is in sybase mode, it performs the following transformation:
  - Removes the currency symbol from money datatypes
  - Transforms quoted strings to quoting conventions specific to the target DBMS

Datatype constants are not transformed in any way except as described previously. When passing datatype constants, the client must verify that the constants are in the proper format required by the target DBMS.

For more information about SQL transformation modes, see Chapter 6, "Issuing SQL Statements."

Data values received as parameters

When an access service receives data values as parameters to RPC commands, cursor commands, or dynamic SQL commands, it converts Open Client and Open Server datatypes to default target DBMS datatypes.

In most cases, Open Client and Open Server datatypes directly map to target datatypes, and the service library defines default mapping rules. However, if the defaults are not valid, the CT-Library client specifies the intended target datatype through the usertype field of the CS_DATAFMT structure.

An Open Server datatype without an associated user-defined datatype is transformed to an ODBC datatype as identified in Table B-1 on page 162.:

The client application can obtain the actual target DBMS datatype for a particular column through the sp_columns CSPs.

The following table shows Open Server datatypes, assigned user-defined datatypes, TDS LAN datatypes, and the resulting ODBC datatypes.

CS_DATAFMT usertype field values

The usertype field of the CS_DATAFMT structure is a 32-bit integer. The client application can specify a target DBMS datatype for a given column. The client application obtains the column datatype indicator from the REMOTE_DATA_TYPE column of the sp_columns result set.

The client application must place the value from sp_columns in the least-significant byte of the usertype field. The remainder of the value is ignored. If a value of 0 is used, the default conversion applies.

The user-defined datatype is used in the child process to transform the Open Server datatype to the ODBC datatype.

Datatype names

An access service receives datatype names as part of create table or alter table commands as follows:

- If the access service is in passthrough mode, the datatype names are not modified.
- If the access service is in sybase mode, Sybase names are assumed and are converted to corresponding target-specific datatype names.

A given target may not be able to support all Open Client and Open Server datatypes, but it permits conversion to an equivalent or compatible datatype. For example, the CS_MONEY datatype can be converted to a numeric (19,4) or equivalent datatype.

For more information about transformation modes, see Chapter 6, "Issuing SQL Statements."

Informix ODBC-supported datatypes

The following table identifies the supported Informix 9 datatypes and their corresponding ODBC datatypes.

Table B-2: Informix datatypes and related ODBC datatypes

Informix 9 datatypes	ODBC datatypes
CHAR	SQL_CHAR(1)
NCHAR	SQL_CHAR(1)
VARCHAR	SQL_VARCHAR(12)
NVARCHAR	SQL_VARCHAR(12)
LVARCHAR	SQL_VARCHAR(12)
*TEXT	SQL_LONGVARCHAR(-1)
*BYTE	SQL_LONGVARBINARY(-4)

Informix 9 datatypes	ODBC datatypes
BOOLEAN	SQL_BIT(-7)
INT8	SQL_BIGINT(-5)
SERIAL8	SQL_BIGINT(-5)
*BLOB	SQL_LONGVARBINARY(-4)
*CLOB	SQL_LONGVARCHAR(-1)
DECIMAL	SQL_DECIMAL(3)
MONEY	SQL_DECIMAL(3)
SMALLINT	SQL_SMALLINT(5)
INTEGER	SQL_INTEGER(4)
SERIAL	SQL_INTEGER(4)
FLOAT	SQL_DOUBLE(8)
SMALLFLOAT (4)	SQL_REAL(7)
DATE	SQL_TYPE_DATE(91)
DATETIME YEAR TO DAY	SQL_TYPE_DATE(91)
DATETIME HOUR TO SECOND	SQL_TYPE_TIME(92)
DATETIME HOUR TO FRACTION(5)	SQL-TYPE_TIME(92)
DATETIME YEAR TO SECOND	SQL_TYPE_TIMESTAMP(93)
INTERVAL YEAR() TO YEAR	SQL_INTERVAL_YEAR(101)
INTERVAL MONTH() TO MONTH	SQL_INTERVAL_MONTH(102)
INTERVAL DAY() TO DAY	SQL_INTERVAL_DAY(103)
INTERVAL HOUR() TO HOUR	SQL_INTERVAL_HOUR(104)
INTERVAL MINUTE() TO MINUTE	SQL_INTERVAL_MINUTE(105)
INTERVAL SECOND() TO SECOND	SQL_INTERVAL_SECOND(106)
INTERVAL SECOND() TO FRACTION(5)	SQL_INTERVAL_SECOND(106)
INTERVAL FRACTION TO FRACTION(5)	SQL_INTERVAL_SECOND(106)
INTERVAL YEAR() TO MONTH	SQL_INTERVAL_YEAR_TO_MONTH(107)
INTERVAL DAY() TO HOUR	SQL_INTERVAL_DAY_TO_HOUR(108)
INTERVAL DAY() TO MINUTE	SQL_INTERVAL_DAY_TO_MINUTE(109)
INTERVAL DAY() TO SECOND	SQL_INTERVAL_DAY_TO_SECOND(110)
INTERVAL DAY() TO FRACTION(5)	SQL_INTERVAL_DAY_TO_SECOND(110)
INTERVAL HOUR() TO MINUTE	SQL_INTERVAL_HOUR_TO_MINUTE(111)
INTERVAL HOUR() TO SECOND	SQL_INTERVAL_HOUR_TO_SECOND(112)
INTERVAL HOUR() TO FRACTION(5)	SQL_INTERVAL_HOUR_TO_SECOND(112)
INTERVAL MINUTE() TO SECOND	SQL_INTERVAL_MINUTE_TO_SECOND(113)
INTERVAL MINUTE() TO FRACTION(5)	SQL_INTERVAL_MINUTE_TO_SECOND(113)

* Results sets containing these datatypes are truncated to 32,768 bytes.

Microsoft SQL Server ODBC-supported datatypes

The following table identifies the supported Microsoft SQL Server datatypes and their corresponding ODBC datatypes.

Table B-3: Microsoft SQL datatypes and related ODBC datatypes

Microsoft SQL Server datatypes	ODBC datatypes
DATE	SQL_TYPE_DATE(91)
CHAR	SQL_WCHAR(1)
NUMERIC	SQL_WNUMERIC(2)
DECIMAL	SQL_WDECIMAL(3)
MONEY	SQL_WDECIMAL(3)
SMALLMONEY	SQL_WDECIMAL(3)
INT	SQL_WINTEGER(4)
SMALLINT	SQL_WSMALLINT(5)
FLOAT	SQL_WFLOAT(6)
REAL	SQL_WREAL(7)
VARCHAR	SQL_WVARCHAR(12)
TEXT	SQL_WLONGVARCHAR(-1)
TIMESTAMP	SQL_WBINARY(-2)
BINARY	SQL_WBINARY(-2)
VARBINARY	SQL_WVARBINARY(-3)
IMAGE	SQL_WLONGVARBINARY(-4)
TINYINT	SQL_WTINYINT(-6)
BIT	SQL_WBIT(-7)
* NCHAR	SQL_WCHAR(-8)
* NVARCHAR	SQL_WVARCHAR(-9)
* NTEXT	SQL_WLONGVARBINARY(-10)
TIMESTAMP	SQL_TYPE_TIMESTAMP(93)
DATETIME	SQL_TYPE_WTIMESTAMP(93)
SMALL DATETIME	SQL_TYPE_WTIMESTAMP(93)

* These datatypes are converted to single-byte ASCII. Future support is anticipated.

DB2 UDB / ODBC-supported datatypes

The following table identifies the supported DB2 datatypes and their related ODBC datatypes.

Table B-4: Supported DB2 datatypes and related ODBC datatypes

DB2 datatypes	ODBC datatypes
CHAR	SQL_CHAR(1)
CHAR () (for bit data)	SQL_BINARY(-2)
DATE	SQL_TYPE_DATE(91)
DECIMAL	SQL_DECIMAL(3)
DOUBLE	SQL_DOUBLE(8)
FLOAT	SQL_FLOAT(6)
FLOAT (4)	SQL_REAL(7)
INTEGER	SQL_INTEGER(4)
* CLOB	SQL_LONGVARCHAR(-1)
LONG VARCHAR	SQL_LONGVARCHAR(-1)
* BLOB	SQL_LONGVARBINARY(-4)
LONG VARCHAR () (for bit data)	SQL_LONGVARBINARY(-4)
NUMERIC	SQL_NUMERIC(2)
SMALLINT	SQL_SMALLINT(5)
TIME	SQL_TYPE_TIME(92)
TIMESTAMP	SQL_TYPE_TIMESTAMP(93)
VARCHAR	SQL_VARCHAR(12)
VARCHAR () (for bit data)	SQL_VARBINARY(-3)

* Results sets containing these datatypes are truncated to 32,768 bytes.

APPENDIX C Using Stored Procedures

This appendix describes how to use stored procedures. It contains the following topics:

Using SQL stored procedures

This section describes how to use SQL stored procedures and covers the following topics:

- Setting up SQL stored procedures
- Running SQL stored procedures

Setting up SQL stored procedures

SQL stored procedures are single SQL statements that are statically bound to the database. Once created, these procedures can be used by any client.

The DirectConnect access service does not support creating SQL stored procedures, because the SQL transformation process does not provide support to handle the translation. You must create SQL stored procedure source code outside of DirectConnect.

The access service does not support either dropping SQL stored procedures or granting authorization for them. Both of these functions are target-dependent.

Running SQL stored procedures

Clients execute SQL stored procedures in different ways, depending upon the SQL transformation mode in effect.

In passthrough mode, the command is as follows:

{call *procname*(*parm1*, *parm2*, ... *parmn*)}

or

{call *procname*(?, ?, ?, ..., ?)}

where *procname* is the name of the procedure.

In sybase mode, the command is as follows:

EXEC *procname argvalues*

or

EXECUTE *procname argvalues*

where:

- *procname* is the name of the procedure.
- argvalues is a list of argument values separated by commas.

The values must be specified in the exact order specified in the create procedure statement contained in the SQL stored procedure. Input parameters are positional.

Datatype for argument values

The datatype for all argument values must be consistent with the datatypes of the relevant columns, as follows:

- Any argument declared as numeric can be used with any numeric column.
- Any argument declared as character can be used with any character column. The access service library treats TEXT as CHAR or VARCHAR.
- Any argument declared as character may be used with any DATE, TIME, or TIMESTAMP column. The value must be in the proper format for the specified type.

Character arguments

Character arguments in sybase mode must be delimited in one of the following ways:

- The value can be enclosed in single or double quotes.
- The value can be enclosed in one of the following special delimiters (passthrough or tsql0 modes only):
    - !
    - %
    - (
    -)
    - *
    - /
    - :
    - <
    - >
    - ?
    - \
    - |
    - ‘
    - {
    - }
    - ~
- The same character must be used before and after the value.

Note Do not enclose numeric arguments in quotes or any special delimiter.

You can specify NULL values in sybase mode in either of the following ways:

- With either “NULL” or “null” used as the value
- With the argument left out of the list (if the procedure contains more than one argument)

Rules for running SQL stored procedures

Use the following rules to run SQL stored procedures:

- You can use binary values in character arguments. The argument cannot contain the 0 value, because the access service library sends all character arguments to ODBC as null terminated strings.
- Do not use escape sequences, because the access service library does not support them. For example, “ABCD\n” is sent as a six-character string.
- Character values sent in the use procedure statement can be longer than declared in the create procedure statement. Any extra characters are truncated, and no error message is sent.
- If a numeric value has a larger scale than that in the target column, the argument is truncated, and no error is recorded.

When a select is issued as a SQL stored procedure, column names are not available. If the client requests column names, the access service library returns dummy names.

Using DB2 stored procedures

This section covers the following topics:

- Description of DB2 stored procedures
- Running DB2 stored procedures

Description of DB2 stored procedures

DB2 stored procedures (external stored procedures) are customer-written programs that reside on the mainframe. The programs can be written in assembler, COBOL, PL/1 or C, and execute within the DB2 stored procedure *ADDRESS* space.

Note DirectConnect for DB2 UDB does not support RSPs or host-resident requests.

Running DB2 stored procedures

Clients execute DB2 stored procedures in different ways, depending upon the SQL transformation mode in effect.

In passthrough mode, the command is as follows:

```
{call procname(parm1, parm2, ... parmn )}
```

or

```
{call procname( ?, ?, ?, ..., ? )}
```

where *procname* is the name of the stored procedure.

In sybase mode, the command is as follows:

```
EXEC procname argvalues
EXECUTE procname argvalues
```

where:

- *procname* is the name of the stored procedure.
- argvalues is a list of argument values separated by commas.

Rules for datatypes as argument values

The datatype for all argument values must be consistent with the datatypes of the relevant columns, as follows:

- Any argument declared as *numeric* can be used with any numeric column.
- Any argument declared as *character* can be used with any character column. The access service library treats TEXT as *CHAR* or *VARCHAR*.
- Any argument declared as *character* may be used with any DATE, TIME, or TIMESTAMP column. The value must be in the proper format for the specified datatype.

APPENDIX D

Using Sybase Mode Commands

This chapter describes the TransacTransact-SQL commands recognized in sybase mode and the corresponding target SQL that the access service library generates.

Transact-SQL commands

Many Transact-SQL commands use table names of up to three parts. ODBC supports the following three-part naming convention:

- *location_name* (DBMS name, current server)
- *authorization_ID* (owner)
- *table_name or view_name*

If the first or second parts are not present, they are omitted. The access service supports three-part objects in all cases in which a SQL object is required in a SQL statement. In doing so, the SQL transformation code either drops the qualifier or performs the correct action with the qualifier.

When the access service receives a use database_name in sybase mode, it captures the name and sets a member of smConnectionConcrete with the database name so that SQL_GETINFO and SQL_DATABASE_NAME pass back the current database name.

The following table lists each command, its restrictions (if any), and its description.

Table D-1: Transact-SQL commands

Command	Restrictions	Description	Page
alter table (core)	core	Adds new columns to an existing table	177
begin transaction	Transact-SQL only	Marks the starting point of a user-defined transaction	179
commit transaction	Transact-SQL only	Marks the ending point of a user-defined transaction	179
create index	core	Creates an index on one or more columns in a table	180
create table	minimum	Creates new tables	182
create view	core	Creates a new view	185
delete	minimum	Removes rows from a table	186
delete (cursor event)	core	Removes a row from a table (if row was made current by a read cursor)	188
delete (dynamic event)	minimum	Removes rows from a table	189
drop index	core	Removes an index from a table in the current database	190
drop table	minimum	Removes a table definition from the database	190
drop view	core	Removes one or more views from the current database	191
execute		Runs a system procedure or user-defined storage procedure	192
grant	core	Assigns authorization to users	192
insert	minimum	Adds new rows to a table or view	194
prepare transaction		Determines whether a server is prepared to commit a transaction	195
revoke	core	Revokes permissions from users	195
rollback transaction		Rolls back a user-specified transaction	198
select	minimum	Retrieves rows from database objects	199
truncate table	extensionusing where 1=1	Removes all rows from a table	202
update	core	Changes existing rows by adding or modifying data	203
update (cursor event)	core	Changes data in a row made current by a read cursor	204

Command	Restrictions	Description	Page
update (dynamic event)	core	Changes data in existing rows of a referenced table	205
use		Specifies the database in which you want to work	207

alter table (core)

Description

This command provides the following functions:

- Adds new columns
- Adds, changes, or drops constraints
- Partitions or unpartitions an existing table

Syntax

Transact-SQL Syntax

```
alter table [database.[owner].]table_name

{add column_name datatype
[default {constant_expression | user | null}]
 {[{identity | null}]
 | [[constraint constraint_name]
 {{unique | primary key}
 [clustered | nonclustered]
 [with {fillfactor | max_rows_per_page} = x]
 [on segment_name]
 [references [[database.]owner.]ref_table
[(ref_column)]
 | check (search_condition)}]}...
 {[, next_column]}...

| add {[constraint constraint_name]
 {unique | primary key}
 [clustered | nonclustered]
 (column_name [{, column_name}...])
 [with {fillfactor | max_rows_per_page} = x]
 [on segment_name]
 | foreign key (column_name [{, column_name}...])
 references [[database.]owner.]ref_table
[(ref_column [{, ref_column}...])]
 | check (search_condition)}

| drop constraint constraint_name

| replace column_name
default {constant_expression | user | null}
| partition number_of_partitions
| unpartition}
```

ODBC Syntax

```
    ALTER TABLE base_table_name
{ADD column_identifier datatype
|ADD(column_identifier datatype[,column_identifier datatype]...)
|DROP[COLUMN]column_identifier[CASCADE|RESTRICT]}
```

Parameters

null
: specifies that you should assign a NULL value when a value is not provided during an insertion.

table_name
: is the name of the table to be changed.

column_name
: is the name of a column to be added.

datatype
: is any of the system datatypes except Bit. If the transformation mode is passthrough, the datatype is expressed as an ODBC datatype.

next_column
: indicates that you can include additional column definitions separated by commas, using the same syntax described for a column definition.

Examples

```
alter table publishers
add manager_name varchar(40) null
```

This adds the *manager_name* column to the publishers table. For each existing table row, a NULL value is assigned to the new column.

Usage

- ASE/CIS sends the alter table command to DirectConnect as a language event.
- The following are not supported:
  - add constraint
  - drop constraint
  - replace column name
  - partition | unpartition
- Transformation adds parentheses when the add column option includes more than one column.

begin transaction (Transact-SQL only)

Description

Marks the starting point of a user-defined transaction.

Syntax

begin tran[saction][*transaction_name*]

Parameters

transaction_name
is the name assigned to the transaction. It must conform to the rules for identifiers. Use transaction names only on the outermost pair of nested begin transaction/commit or begin transaction/rollback statements.

Examples

```
begin transaction
```

Usage

- The access service library accepts the transaction name parameter, then strips it before passing it to the target.
- The begin transaction command is not recognized by ODBC, so DirectConnect traps this statement and monitors the transaction state internally. When a commit is issued after a begin transaction, SQL Transact() is called.
- Only one transaction can be open per connection.

commit transaction (Transact-SQL only)

Description

Marks the ending point of a user-defined transaction.

Syntax

commit [tran[saction] | work][*transaction_name*]

Parameters

transaction_name
is the name assigned to the transaction. It must conform to the rules for identifiers. Use transaction names only on the outermost pair of nested begin transaction/commit or begin transaction/rollback statements.

Examples

```
commit transaction
```

Usage

- The access service strips the *transaction_name* from the statement before passing it on the target.
- *transaction_name* is not used with version 10.5 of ASE/CIS.
- The commit command is not recognized by ODBC, so DirectConnect traps this statement and monitors the transaction state internally. When a commit is issued after a begin transaction, SQL Transact() is called.
- Only one transaction can be open per connection.

create index (core)

Description

Creates an index on one or more columns in a table.

Syntax

Transact-SQL Syntax

create [unique][clustered | nonclustered]
 index *index_name*

on [[*database*.]*owner*.]*table_name*(*column_name*
[, *column_name*]...)

[with {{fillfactor | *max_rows_per_page*} = x, *ignore_dup_key*, *sorted_data*,
 [*ignore_dup_row* | *allow_dup_row*]}]

[on *segment_name*]

ODBC Syntax

CREATE [UNIQUE] INDEX *index_name*

ON *base_table_name*

(*column_identifier*[ASC|DESC]
 [,*column_identifier*[ASC|DESC]]...)

Parameters

unique
: prohibits duplicate index values (key values). The system checks for duplicate key values when an index is created and checks each time data is added with an insert or update. If a duplicate key value exists, or if more than one row contains a NULL value, the command aborts and an error message shows the duplicate value prints.

clustered
: indicates that the physical order of rows on this table is the same as the indexed order of rows. Only one clustered index per table is permitted.

nonclustered
: indicates that a level of indirection exists between the index structure and the data. Up to 249 nonclustered indexes per table are permitted.

fillfactor
: specifies how full the DBMS makes each page when it creates a new index on existing data. This percentage is relevant only at the time the index is created. As the data changes, the pages are not maintained at any level of fullness. The default is 0. If the fillfactor is set to 100, the DBMS creates indexes with pages 100% full.

ignore_dup_key
: responds to a duplicate key entry into any table with a unique index. An attempted insert of a duplicate key is ignored, and the insert is canceled with an informational message.

on segment_name
: specifies that the index is to be created on the named segment.

index_name
: is the name of the index. Index names must be unique within a table but need not be unique within a database.

table_name
: is the name of the table that contains the indexed column or columns.

column_name
: is the column or columns to be included in the index. Composite indexes are based on the combined values of up to 16 columns. The sum of the maximum lengths of all the columns used in a composite index cannot exceed 256 bytes.

Examples

```
create index au_id_ind
on authors (au_id)
```

```
create index ind1
 on titleauthor (au_id, title_id)
```

```
create nonclustered index zip_ind
on authors (zip) with fillfactor = 25
```

Usage

- ASE/CIS sends the create index command to DirectConnect as a language event.
- Columns of type bit, text, and image cannot be indexed.
- You cannot create an index on a view.
- The following parts of the create index command are not recognized by transformation:
    - clustered | nonclustered
    - with fillfactor
    - *max_rows_per_page*
    - *ignore_dup_key*
    - *sorted_data*
    - *ignore_dup_row* | *all_dup_row*
- on segment

- The ODBC command ASC|DESC cannot be generated by transformation. Only ascending indexes can be created.

create table (minimum)

Description

Creates new tables and optional integrity constants.

Syntax

Transact-SQL Syntax

```
create table [database.[owner.]table_name (column_name datatype
[default {constant_expression | user | null}]
 {[{identity | null | not null}]
 | [[constraint constraint_name]
 {{unique | primary key}
 [clustered | nonclustered]
 [with {fillfactor | max_rows_per_page} = x]
 [on segment_name]
 | references [[database.]owner.]ref_table
[(ref_column)]
 | check (search_condition)}]}...
```

```
| [constraint constraint_name]
 {{unique | primary key}
 [clustered | nonclustered]
 (column_name [{, column_name}...])
 [with {fillfactor | max_rows_per_page} = x]
 [on segment_name]
 | foreign key (column_name [{, column_name}...])
 references [[database.]owner.]ref_table
[(ref_column [{, ref_column}...])]
 | [check (search_condition)}
```

```
[{, {next_column | next_constraint}}...])
```

```
[with max_rows_per_page = x][on segment_name]
```

ODBC Syntax

```
CREATE TABLE base_table_name
(column_element[,column_element]...)
```

```
column_element::=column_definition|
 table_constraint_definition
```

```
column_definition:=
 column_identifier datatype
[DEFAULT default_value]
 [column_constraint_definition
[column_constraint_definition]...]
```

default_value::=literal|NULL|USER

column_constraint_definition::=
 NOT NULL
 |UNIQUE|PRIMARY KEY
 |REFERENCES *ref_table_name referenced_columns*

table_constraint_definition::=
 UNIQUE(*column_identifier*[,*column_identifier*]...)
 |PRIMARY KEY(*column_identifier*[,*column_identifier*]...]
 |CHECK(*search_condition*)
 |FOREIGN KEY *referencing_columns* REFERENCES
 ref_table_name referenced_columns

Parameters

null | not null
: specifies a NULL value if you do not provide a value during an insertion and no default exists (for null), or that you must provide a non-NULL value if no default exists (for not null).

next_column
: indicates that you can include additional column definitions (separated by commas) using the same syntax described for a column definition.

on segment_name
: specifies the name of the segment on which to place the table.

table_name
: is the name of the new table. It conforms to the rules for identifiers and is unique within the database and to the owner.

column_name
: is the name of the column in the table. It conforms to the rules for identifiers and is unique in the table.

datatype
: is the datatype of the column. Only system datatypes are used. As shown in Table D-2 on page 184, several datatypes expect a length, *n*, in parentheses: *datatype*(*n*).

Examples

```
create table titles
    (title_id tid not null,
    title varchar(80) not null,
    type char(12) not null,
    pub_id char(4) null,
    price money null,
    advance money null,
    total_sales int null,
    notes varchar(200)null,
    pubdate datetime not null,
    contract bit not null)
```

Creates the *titles* table.

Usage

- The following Transact-SQL parts of the create table command are not recognized by sybase transformation mode:
    - with fillfactor
    - clustered | nonclustered
    - with *max_rows_per_page*
    - on segment name
- Transact-SQL allows you to specify null or not null, with a default of not null. ODBC allows only not null to be specified. The default is null.

The following table shows how the access service transforms this clause.

Table 12-7: Null transformations during Transact-SQL to ODBC CREATE TABLE

Transact-SQL specification	Transformed to
null	null
not null	not null
<nothing>	not null

Table D-2 shows how the access service transforms Transact-SQL datatypes. The selection order is as follows:

- The access service first attempts to change the Transact-SQL datatype to the primary ODBC datatype.
- If the ODBC driver does not support the ODBC datatype, then the access service uses the secondary ODBC datatype.
- If the secondary ODBC datatype is likewise unsupported, the access service uses the final ODBC datatype.

Because the ODBC driver may not support extended datatypes such as Tinyint and Bit, a core ODBC type is associated with those datatypes as the second and third choices.

Table D-2: Datatype conversions of Transact-SQL to ODBC

Transact-SQL datatype	Primary ODBC datatype	Secondary ODBC datatype	Final ODBC datatype
Tinyint	SQL_TINYINT	SQL_SMALLINT	SQL_INTEGER
Smallint	SQL_SMALLINT	SQL_INTEGER	SQL_INTEGER
Int	SQL_INTEGER	SQL_INTEGER	SQL_INTEGER
Numeric	SQL_NUMERIC	SQL_DECIMAL	SQL_FLOAT

Transact-SQL datatype	Primary ODBC datatype	Secondary ODBC datatype	Final ODBC datatype
Decimal	SQL_DECIMAL	SQL_NUMERIC	SQL_FLOAT
Float	SQL_FLOAT	SQL_DOUBLE	SQL_CHAR
Double Precision	SQL_DOUBLE	SQL_FLOAT	SQL_FLOAT
Real	SQL_REAL	SQL_FLOAT	SQL_FLOAT
Smallmoney	SQL_DECIMAL	SQL_NUMERIC	SQL_FLOAT
Money	SQL_DECIMAL	SQL_NUMERIC	SQL_FLOAT
Smalldatetime	TIMESTAMP	SQL_CHAR	SQL_CHAR
Datetime	TIMESTAMP	SQL_CHAR	SQL_CHAR
Char	SQL_CHAR	SQL_CHAR	SQL_CHAR
Varchar	SQL_VARCHAR	SQL_VARCHAR	SQL_VARCHAR
Nchar	1 (SQL_CHAR(2n))	SQL_CHAR	SQL_CHAR
Nvarchar	12 (SQL_VARCHAR(2n))	SQL_VARCHAR	SQL_VARCHAR
Text	SQL_LONGVARCHAR	SQL_VARCHAR	SQL_VARCHAR
Binary	SQL_BINARY	SQL_VARBINARY	SQL_CHAR
Varbinary	SQL_VARBINARY	SQL_LONGVARBINARY	SQL_VARCHAR
Image	SQL_LONGVARBINARY	SQL_LONGVARCHAR	SQL_VARCHAR
Bit	SQL_BIT	SQL_CHAR	SQL_CHAR

The size of the *char* value for Nchar and Nvarchar conversions should be doubled to accommodate the double-byte characters possible.

For Text secondary and final ODBC datatype values, and Image final ODBC datatype values, the original value can be truncated to fit the “transformed to” value.

Binary, Varbinary, and Image final ODBC datatype values can go through code set translation, which is not desirable for binary datatypes.

create view (core)

Description

Creates a new view.

Syntax

Transact-SQL Syntax

```
create view [owner.]view_name
[(column_name [, column_name]...)]
 as select [distinct] select_statement
[with check option]
```

ODBC Syntax

```
CREATE VIEW viewed_table_name
[(column_identifier[,column_identifier]...)]
AS query_specification
```

Parameters

select
: begins the select statement that defines the view.

distinct
: specifies that the view cannot contain duplicate rows (optional).

with check option
: indicates that all data modification statements are validated against the view selection criteria. All rows inserted or updated through the view must remain visible through the view.

view_name
: is the name of the view. The view name cannot include the database name. It must conform to the rules for identifiers.

column_name
: is the name of the column in the view. It must conform to the rules for identifiers.

select_statement
: completes the select statement that defines the view. It can include more than one table and other views.

Examples

```
create view “new view” (“column 1”, “column 2”)
 as select col1, col2 from “old view”
```

In this example, the *new view* is created from the *old view*.

Both columns are renamed in the new view. All view and column names that include embedded blanks are enclosed in double quotation marks. Before creating the view, you must set the quoted_identifier property to on.

Usage

- You can use views as security mechanisms by granting authorization on a view but not on its underlying tables.
- The with check option is removed from the transformed SQL text.

delete (minimum)

Description

Removes rows from a table.

Syntax

Transact-SQL Syntax

```
delete [from][[database.]owner.]{view_name | table_name}
 [where search_conditions]
```

```
delete [[database.]owner.]{table_name | view_name}
 [from [[database.]owner.]{view_name | table_name
[(index index_name [prefetch size][lru | mru])]}
 [, [[database.]owner.]{view_name | table_name
(index index_name [prefetch size][lru | mru])]} ]...]
 [where search_conditions]
```

ODBC Syntax

```
DELETE FROM table_name [WHERE search_condition]
```

Parameters

from
: (after delete) allows you to name more than one table or view to use with a where clause when specifying the rows to delete. The from clause allows you to delete rows from one table based on data stored in other tables, giving you much of the power of an embedded select statement.

from
: (after *table_name* or *view_name*) is an optional keyword used for compatibility with other versions of SQL. Follow it with the name of the table or view from which you want to remove rows.

where
: is a standard where clause.

where current of
: *cursor_name* causes Adaptive Server to delete the table row or view indicated by the current cursor position for *cursor_name*.

Note Since changing the name "Sybase SQL Server®" to "Adaptive Server Enterprise," Sybase uses the names "Adaptive Server Enterprise" and "Adaptive Server" to refer to all supported versions of Sybase SQL Server and Adaptive Server Enterprise. Version-specific references to Adaptive Server or SQL Server include version numbers.

Examples

```
delete from authors
 where au_lname = "McBadden"
```

Usage

- You cannot use delete with a multi-table view.
- If you do not use a where clause, all rows are deleted from the table or view that is named after the delete [from] Transact-SQL keyword parameter.

- Upon completion of the delete command, the number of rows affected must be indicated.

delete (core)

Description

Removes a row from a table (cursor event). The affected row must have been made current by a read cursor.

Syntax

Transact-SQL Syntax

```
delete [from][[database.]owner.]{table_name | view_name}
 where current of cursor_name
```

ODBC Syntax

```
DELETE FROM table_name WHERE CURRENT OF cursor_name
```

Examples

```
delete titles where current of title_crsr
```

Usage

- ASE/CIS (formerly OmniConnect™) issues a cursor delete request if it examines any column data to fulfill the client request. This is true for the following:
    - More than one table is involved in the delete statement.
    - The statement contains built-in functions in the where clause.
- ASE/CIS passes the delete command to DirectConnect as the following series of cursor commands:
    - declare
    - open
    - close
    - deallocate

  The where clause is not constructed. Open Server appends the equivalent of where current of cursor read_cursor_name.
- The cursor can be reused multiple times before it is deallocated.
- When DirectConnect calls srv_senddone to mark the completion of the delete command, the number of affected rows must be indicated. Normally, the row count is 1.
- Any valid object in the catalog can be substituted for *table_name* or *view_name*.

delete (minimum)

Description

Removes rows from a table (dynamic event).

Syntax

Transact-SQL Syntax

```
delete [[database.]owner.]{table_name | view_name}
 [where column_name relop ? {AND | OR} column_name relop ? ...]]
```

Parameters

relop

is a relational operation.

?

is the parameter marker.

Usage

- ASE/CIS issues a dynamic delete request if it does not have to examine any column data to fulfill the client request. This is true for the following:
  - Only one table is involved in the delete statement.
  - The statement contains no built-in functions in its where clause.
- Supported relational operators are:

  =, <>, <, >, <=, >=, LIKE.
- ASE/CIS passes the dynamic delete command to DirectConnect as a series of dynamic requests:
  - prepare
  - define (parameter formats)
  - execute (with parameter data)
  - deallocate
- The where clause is optional. It is provided by ASE/CIS if the original delete command contained one.
- The prepared statement can execute multiple times before it is deallocated.
- Upon completion of the delete command, the number of affected rows must be indicated.
- Any valid object in the catalog can be substituted for *table_name*, *view_name*, and other variables.

drop index (core)

Description

Removes an index from a table in the current database.

Syntax

Transact-SQL Syntax

```
drop index table_name.index_name
[, table_name.index_name]...
```

ODBC Syntax

```
DROP INDEX index_name
```

Parameters

table_name
is the table in which the indexed column is located. The table must be in the current database.

index_name
is the name of the index to be dropped.

Examples

```
drop index authors.au_id_ind
```

Usage

- You can specify multiple index names in Transact-SQL, but ODBC supports only a single name per statement. If multiple names are present, multiple ODBC DROP statements are generated.
- ASE/CIS passes the drop index command to DirectConnect as a language event.
- To get information about existing indexes on a table, use:

  ```
  sp_helpindex table_name
  ```

drop table (minimum)

Description

Removes a table definition and all of its data, indexes, triggers, and permissions from the database.

Syntax

Transact-SQL Syntax

```
drop table [[database.]owner.]table_name
[, [[database.]owner.]table_name]...
```

ODBC Syntax

```
DROP TABLE base_table_name
[CASCADE|RESTRICT]
```

Parameters

table_name
is the name of the table to be dropped.

Examples

```
drop table authors
```

Usage

- Transact-SQL allows you to drop multiple tables in one statement, but ODBC does not. If multiple tables are encountered, transformation generates multiple DROP statements.
- ODBC allows the keywords CASCADE and RESTRICT to be used in the statement. Since Transact-SQL does not support this, the keywords are not generated during transformation.
- ASE/CIS passes the drop table command to DirectConnect as a language event.

drop view (core)

Description

Removes one or more views from the current database.

Syntax

Transact-SQL Syntax

drop view [*owner*.]*view_name*[,[*owner*.]*view_name*]...

ODBC Syntax

DROP VIEW *viewed_table_name*
[CASCADE|RESTRICT]

Parameters

view_name
is the name of the view to be dropped. The name must be a legal identifier and cannot include a database name.

Examples

drop view *new_price*

This removes the view *new_price* from the current database.

Usage

- Transact-SQL allows you to drop multiple views in one statement, but ODBC does not. If multiple views are encountered, transformation generates multiple DROP statements.
- ODBC allows the keywords CASCADE and RESTRICT to be used in the statement. Since Transact-SQL does not support this, these keywords are not generated during transformation.
- Each time a view is referenced, another view or stored procedure checks the existence of the view.

execute

Description

Runs a system procedure or a user-defined stored procedure.

Syntax

Transact-SQL Syntax

```
[execute][@return_status = ]
[[[server.]database.]owner.]procedure_name[; number]
 [[@parameter_name =] value |
 [@parameter_name =] @variable [output]
 [,[@parameter_name =] value |
 [@parameter_name =] @variable [output]...]]
[with recompile]
```

ODBC Syntax

```
{CALL PROCEDURE(parm1_value, parm2_value, ..., parmN_value)}
```

Usage

- This transformation uses only the shorthand notation ODBC syntax.
- Procedure return values are not supported.
- Procedures that return multiple result sets are not supported.

grant (core)

Description

Assigns authorization to users.

Syntax

Transact-SQL Syntax

To grant authorization to access database objects:

```
grant {all [privileges] | permission_list}
 on {table_name [(column_list)]
 | view_name[(column_list)]
 | stored_procedure_name}
 to {public | name_list | role_name}
 [with grant option]
```

To grant authorization to create database objects:

```
grant {all [privileges] | command_list}
 to {public | name_list | role_name}
```

ODBC Syntax

```
GRANT {ALL|grant_privilege[,grant_privilege]...}
ON table_name
TO {PUBLIC|user_name[,user_name]...}
```

```
grant privilege::=
 DELETE
 | INSERT
 | SELECT
 | UPDATE[(column_identifier[,column_identifier]...)]
 | REFERENCES[(column_identifier[,column_identifier]
 ...)]
```

Parameters

all
: when used to assign authorization to access database objects (first syntax format), specifies that all privileges applicable to the specified object are granted or revoked.

public
: is all users of the "public" group, which includes all users of the system.

with grant option
: allows the users specified in *name_list* to grant the privileges specified by *permission_list* to other users.

permission_list
: is a list of authorizations granted.

command_list
: is a list of commands granted.

table_name
: is the name of a table in the database.

column_list
: is a list of columns, separated by commas, to which the privileges apply.

view_name
: is the name of a view in the current database. Only one view can be listed for each grant command.

stored_procedure
: is the name of a stored procedure in the database.

name_list
: is a list of user database names or group names or both, separated by commas.

role_name
: is the name of a SQL Server role. Use it to grant authorizations to all users who have been granted a specific role.

Examples

```
grant insert, delete
 on titles
 to mary, sales
```

```
grant update
 on titles (price, advance)
 to public
```

```
grant create database, create table
 to mary, john
```

```
grant update on authors
 to mary
 with grant option
```

Usage

- Any valid object in the catalog can be substituted for *table_name* or *view_name*.
- with grant option is not available. Transformation removes this phrase.
- ODBC does not allow you to grant authorization to a stored procedure.
- You can substitute from for to in the grant syntax.
- You can grant or revoke authorizations only on objects in the current database.

insert (minimum)

Description

Adds new rows to a table or view.

Syntax

Transact-SQL Syntax

```
insert [into][database.[owner.]]{table_name | view_name}
 [(column_list)]
 {values (expression [, expression]...)
 | select_statement}
```

ODBC Syntax

```
INSERT INTO table_name[(column_identifier[,column_identifier]
 ...)]
```

{*query_specification*|VALUES(*insert_value*
[,*insert_value*]...)

Parameters

values
: is a keyword that introduces a list of expressions.

?
: specifies parameters passed by ASE/CIS at the time the insert command is executed.

column_list
is a list of one or more columns to which data is to be added. The columns can be in any order, but the incoming data (whether in a values clause or select clause) is in the same order.

Examples

```
insert titles
 (title_id, title, type, pub_id, notes, pubdate,contract)
 values (?, ?, ?, ?, ?, ?, ?)
```

Usage

- Any valid object in the catalog can be substituted for *table_name*, *view_name*, and so forth.
- ASE/CIS passes the insert command to DirectConnect as the following series of dynamic SQL commands:

  prepare

  execute

  close

  deallocate
- The values in the values list are passed as dynamic SQL parameters.

prepare transaction

Description
Used by a DB-Library in a two-phase commit application to determine whether a server is prepared to commit a transaction.

Syntax
Transact-SQL Syntax

```
prepare tran[saction]
```

ODBC Syntax

Not supported.

Usage
Not supported

revoke (core)

Description
Revokes permissions from users.

Syntax
Transact-SQL Syntax

To revoke permission to access database objects:

```
revoke [grant option for]
 {all [privileges] | permission_list}
 on {table_name [column_list)]
 | view_name [(column_list)]
 | stored_procedure_name}
 from {public | name_list | role_name}
 [cascade]
```

To revoke permission to create database objects:

```
revoke {all [privileges] | command_list}
 from {public | name_list | role_name}
```

ODBC Syntax

```
REVOKE {ALL|revoke_privilege[,revoke_privilege]...}
ON table_name
FROM {PUBLIC|user_name[,user_name]...}
[CASCADE|RESTRICT]
revoke_privilege::=
 DELETE
 |INSERT
 |SELECT
 |UPDATE
 |REFERENCES
```

This statement revokes authorization from users.

Parameters

all
specifies that all privileges applicable to the specified object are revoked when used to revoke authorizations to access database objects (first syntax format).

public
is all users of the “public” group, which includes all system users.

grant option for
prohibits the users specified in *name_list* from granting the privileges specified by *permission_list* to other users.

cascade
: revokes grant authorization for the privileges specified in *permission_list* from the users specified in *name_list* and from all users to whom they granted privileges.

  The cascading effect occurs even if it is not specified by the user. For example, suppose UserA has granted UserB privileges, and in turn, UserB granted privileges to UserC. If UserA is revoked, all privileges that UserA granted to UserB and UserB indirectly granted to UserC are revoked.

permission_list
: is a list of authorizations to be revoked.

command_list
: is a list of commands for which authorizations are to be revoked.

table_name
: is the name of a table in the database.

column_list
: is a list of columns, separated by commas, to which the privileges apply. If columns are specified, only select and update authorizations can be revoked.

view_name
: is the name of a view in the current database. Only one view can be listed for each revoke statement.

stored_procedure
: is the name of a stored procedure in the database. Only one object can be listed for each revoke statement.

name_list
: is a list of user database names and group names, separated by commas.

role_name
: is the name of an Adaptive Server role. This allows you to revoke from all users who have been granted a specific role.

Examples

```
revoke insert, delete
 on titles
 from mary, sales
```

```
revoke update
 on titles (price, advance)
 from public
```

```
revoke create database, create table
 from mary, john
```

```
revoke execute on new_sproc
from oper_role
```

Usage

- Valid permissions for Transact-SQL are:
  - select
  - insert
  - delete
  - update
  - references
- ODBC does not support revoking a stored procedure.
- Authorizations can be revoked only on objects in the current database.
- grant and revoke commands are order-sensitive. When a conflict occurs, the most recently issued command takes effect.
- to can be substituted for from in the revoke syntax.

rollback transaction

Description

Rolls back a user-specified transaction to the last savepoint inside the transaction or to the beginning of the transaction.

Syntax

Transact-SQL Syntax

```
rollback {tran[saction] | work}
 [transaction_name | savepoint_name]
```

ODBC Syntax

ODBC does not support rollback as a SQL command. DirectConnect traps the command and monitors the transaction state with the child process. If a transaction is opened, a call to SQL Transact() is generated.

Parameters

transaction_name
is the name assigned to the transaction. It must conform to the rules for identifiers.

Examples

```
rollback transaction
```

Usage

- The *transaction name* and *savepoint name* are ignored. Only one pending transaction is allowed for each connection.
- *transaction_name* is not used with ASE/CIS release 10.5.

- The access service strips the *transaction_name* from the statement before passing it to the target.

select (minimum)

Description

Retrieves rows from database objects.

Syntax

Transact-SQL Syntax

```
select [all | distinct] select_list
 [into [[database.]owner.]table_name]
 [from [[database.]owner.]{view_name | table_name
[(index index_name [prefetch size][lru | mru])]}
 [holdlock | noholdlock] [shared]
 [,[[database.]owner.]{view_name | table_name
[(index index_name [prefetch size][lru | mru])]}
 [holdlock | noholdlock] [shared]]...]
[where search_conditions]
[group by [all]aggregate_free_expression
[, aggregate_free_expression]...]
 [having search_conditions]
[order by
 {[[[database.]owner.]{table_name. | view_name.}]
 column_name | select_list_number | expression}
 [asc | desc]
 [,{[[[database.]owner.]{table_name. | view_name.}]
 column_name | select_list_number | expression}
 [asc | desc]]...]
[compute row_aggregate(column_name)
 [, row_aggregate(column_name)]...
 [by column_name [, column_name]...]]
[for {{read only | update [of column_name_list]}}]
[at isolation {read uncommitted | read committed | serializable}]
[for browse]
```

ODBC Syntax

```
SELECT [ALL|DISTINCT]select_list
FROM table_reference_list
[WHERE search_condition]
[GROUP BY column_name[,column_name]...]
```

[HAVING *search_condition*]

[UNION [ALL]*select_statement*]...
 [*order_by_clause*]

An alternate syntax for updating tables if the driver supports core or extended functionality:

SELECT [ALL|DISTINCT]*select_list*

FROM *table_reference_list*

[WHERE *search_condition*]

FOR UPDATE OF [*column_name*[,*column_name*]...]

Parameters

all
includes all rows in the result.

from
indicates a comma-separated list of tables or views to use in the select statement.

group by
finds a value for each group. These values appear as new columns in the results, rather than as new rows.

order by
sorts the results by columns.

having
sets conditions for the group by clause, similar to the way that where sets conditions for the select clause. No limit exists for the number of conditions that can be included.

union
returns a single result set that combines the results of two or more queries. Duplicate rows are eliminated from the result set unless the all keyword is specified.

read only
indicates that the cursor is a read-only cursor and that updates cannot be applied to rows made current by it.

update
indicates that the cursor is an updatable cursor, and that the rows it makes current can be deleted or updated.

select_list
is one or more of the following items:

- A list of column names in the order in which you want them returned
- An aggregate function
- Any combination of these items

table_name
and *view_name* list tables and views used in the select statement.

If more than one table or view is in the list, the names are separated by commas. Table names and view names are given correlating names. You can do this by providing the table or view name, then a space, then the correlation name, such as:

```
select * from publishers t1, authors t2
```

search_conditions
sets the conditions for the rows that are retrieved. A search condition can include column names, constants, joins, the keywords is null, is not null, or, like, and, or any combination of these items.

Examples

```
select count(*) from publishers for read only
```

select *pub_id*, *pub_name*, city, state from publishers for read only

```
select type, price from titles
 where price > @p1 for update of price
```

select *stor_id*, *stor_name* from sales union
 select *stor_id*, *stor_name* from *sales_east*

Usage

- You can issue this command as a language command or a client-based cursor request.
- This statement is accepted and sent to ODBC without change, subject to the qualifications listed in this section.
- The TEXTPTR() function cannot appear in the select list.
- The following SQL Server 10.x aggregate functions are supported:
  - sum ([all | distinct])
  - avg ([all | distinct])
  - count ([all | distinct])
  - count (*)
  - max *(expression)*)
  - min *(expression)*)

- If the command is issued as a cursor, the access service library supports the following cursor commands:
  - declare
  - open
  - fetch
  - close
  - deallocate
- If a cursor is passed a new set of parameters before it is opened, it can be reused multiple times.
- The data values used in the where clause search conditions are passed as cursor parameters, using the datatype associated with the column.
- Cursor parameters are indicated with the "@" character when Transact-SQL syntax is used, and with a question mark when passthrough mode is used.
- The following are not supported:
  - Transact-SQL select into syntax
  - The use of index in a from clause
  - The use of prefetch size in a from clause
  - The use of holdlock|noholdlock|shared in a from clause
  - The "compute" phrase
  - The "at isolation" phrase
  - The "for browse" phrase
- The availability of the GROUP BY, HAVING, and UNION clauses depends upon the ODBC driver level of conformance.

truncate table (extension using where 1=1)

Description

Removes all rows from a table.

Syntax

Transact-SQL Syntax

truncate table [[*database*.]*owner*.]*table_name*

ODBC Syntax

Not supported.

The command can be transformed into an equivalent delete command as follows:

```
DELETE FROM table_name
```

Parameters

table_name
is the name of the table to be truncated.

Examples

```
truncate table authors
```

Usage

ASE/CIS passes the command to DirectConnect as a language event.

update (core)

Description

Changes existing rows by adding data or modifying existing data.

Syntax

Transact-SQL Syntax

```
    update [[database.]owner.]{table_name | view_name}
set [[[database.]owner.]{table_name. | view_name.}]
 column_name1 =
 {expression1 | NULL | (select_statement)}
 [, column_name2 =
 {expression2 | NULL | (select_statement)}]...
[from [[database.]owner.]{view_name | table_name
[(index index_name [prefetch size][lru | mru])]}
 [, [[database.]owner.]{view_name | table_name
[(index index_name [prefetch size][lru | mru])]}]...]
 [where search_conditions]
```

ODBC Syntax

```
    UPDATE table_name
SET column_identifier={expression|NULL}
 [,column_idenfifier={expression|NULL}]...
[WHERE search_condition]
```

Parameters

set
specifies the column name and assigns the new value. The value can be an expression or a NULL. More than one column name or value pair must be separated by commas.

where
: is a standard where clause.

Examples

```
update authors
 set au_lname = "MacBadden"
 where au_lname = "McBadden"
```

Usage

- Use update to change values in rows that have already been inserted. Use insert to add new rows.
- You cannot update views defined with the distinct clause.
- ODBC does not support a from clause in the update statement. The access service ignores the from clause.
- select statements are not supported.
- The "index" phrase is not supported.

update (core)

Description

Changes data in a row made current by a read cursor by adding data or modifying existing data (cursor event).

Syntax

Transact-SQL Syntax

```
update [[database.]owner.]{table_name | view_name}
set [[[database.]owner.]{table_name. | view_name.}]
 column_name1 =
 {expression1 | NULL | (select_statement)}
 [, column_name2 =
 {expression2 | NULL | (select_statement)}
 where current of cursor_name
```

ODBC Syntax

```
UPDATE table_name
SET column_idenfifier={expression|NULL}
 [,column_identifier={expression|NULL}]...
WHERE CURRENT OF cursor_name
```

Parameters

set
: specifies the column name and assigns the new value. The value is passed as a cursor parameter.

where current of
causes Adaptive Server to update the row of the table or view indicated by the current cursor position for *cursor_name*.

Examples

```
update authors
set au_lname = @p1
```

The row made current by the cursor *authors_cursor* is modified. The column *au_lname* is set to the value of the parameter @p1.

Usage

- Any valid object in the catalog can be substituted for *table_name*, *view_name*, and so forth.
- ASE/CIS issues a cursor update request if it must examine any column data to fulfill the client request, subject to the following conditions:
    - More than one table is involved in the update statement.
    - The statement contains built-in functions in its where clause.
    - A column name is referenced to the right of any set expression.
- ASE/CIS passes the update command to DirectConnect as the following series of cursor commands:

  declare (define parameter formats)

  open (define parameter data)

  close

  deallocate
- The cursor can be reused multiple times before it is deallocated.
- Upon completion of the update command, the number of rows affected must be indicated.
- ODBC does not support a from clause in the update statement. The access service ignores the from clause.

update (core)

Description

Changes data in existing rows of the referenced table (dynamic event).

Syntax

Transact-SQL Syntax

```
update [[database.]owner.]{table_name | view_name}
 set column_name1 = ?
 [, column_name2 = ?]...
 [ where column_name relop ?
 [ {AND | OR} column_name relop ? ...]]
```

Parameters

set
: specifies the column name and assigns the new value. The value is passed as a parameter.

relop
: is a relational operation.

Examples

```
update authors
 set au_lname = ?
 where au_id = ?
```

The *au_lname* column is set to the value of <*parameter 1*> where the value of *au_id* is equal to the value of <*parameter 2*>.

Usage

- You can substitute *table_name* and *view_name* with any valid object in the catalog.
- ASE/CIS issues a dynamic update request if it does not need to examine any column data to fulfill the client request, subject to the following conditions:
    - Only one table is involved in the update statement.
    - The statement does not contain built-in functions in its where clause.
    - A column name is not referenced to the right of any set expression.
- The following relational operators (relops) are supported in search conditions:

  =, <>, <, >, <=, >=, LIKE
- ASE/CIS passes the update command to DirectConnect as the following series of dynamic requests:

  prepare (define parameter formats)

  execute (define parameter data)

  deallocate
- Upon completion of the update command, the number of rows affected must be indicated.
- ODBC does not support a from clause in the update statement. The access service ignores the from clause.

use

Description	Specifies the database with which you want to work.
Syntax	*Transact-SQL Syntax* use *database_name* *ODBC Syntax* No ODBC SQL equivalent.
Parameters	database_name is the name of the database you want to access.
Usage	Supported only for drivers and targets that support ODBC's SQL_CURRENT_QUALIFIER connection option. For these targets, the current connection is dropped, and a new connection is opened with SQL_CURRENT_QUALIFIER set to *database_name*.

Glossary

access management	A DirectConnect feature that provides connectivity to non-Sybase targets.
access service	The named set of properties, used with a DirectConnect Access Service Library, to which clients connect. Each DirectConnect server can have multiple services.
Access Service Library	A service library that provides access to non-Sybase data contained in a database management system or other type of repository. Each such repository is called a "target." Each access service library interacts with exactly one target and is named accordingly. See also **service library**.
ACSLIB	See **Access Service Library**.
Adaptive Server	The server in the Sybase Client-Server architecture. It manages multiple databases and multiple users, tracks the actual location of data on disks, maintains mapping of logical data description to physical data storage, and maintains data and procedure caches in memory.
Administrative Service Library	A service library that provides remote management capabilities and server-side support. It supports a number of remote procedures (invoked as RPC requests) that enable remote DirectConnect management. See also **remote procedure call** and **service library**.
ADMLIB	See **Administrative Service Library**.
American Standard Code for Information Interchange	The standard code used for information interchange among data processing systems, data communication systems, and associated equipment. The code uses a coded character set consisting of seven-bit coded characters (eight bits including a parity check).
API	See **application program interface**.
application program interface	A functional interface, supplied by an operating system or other licensed program, that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.
ASCII	See **American Standard Code for Information Interchange**.

ASE/CIS	Adaptive Server Enterprise / Component Integration Services (formerly OmniConnect). An add-on product for Adaptive Server that provides a Transact-SQL interface to external data sources, including host data files and tables in other database systems. OmniConnect replaces OmniSQL Gateway and OmniSQL Server.
bulk copy transfer	A transfer method in which multiple rows of data are inserted into a table in the target database. See also **transfer**. Compare with **destination-template transfer**.
call level interface	A programming style that calls database functions directly from the top level of the code. Usually it is contrasted with embedded SQL. See also **dynamic SQL** and **embedded SQL**.
catalog	A system table that contains information about objects in a database, such as tables, views, columns, and authorizations.
catalog stored procedure	A stored procedure that provides information about tables, columns, and authorizations. It is used in SQL generation and application development. See also **stored procedures**.
character set	A set of specific (usually standardized) characters with an encoding scheme that uniquely defines each character. ASCII is a common character set.
client	In client/server systems, the part of the system that sends requests to servers and processes the results of those requests. See also **client/server**. Compare with **server**.
client application	Software that is responsible for the user interface, including menus, data entry screens, and report formats. See also **Client-Library**.
Client-Library	A library of routines that is part of Open ClientConnect™. See also **Open ClientConnect**.
client/server	An architecture in which the client is an application that handles the user interface and local data manipulation functions, while the server provides data processing access and management for multiple clients. See also **client**, **client application**, and **server**.
clustered index	An index in which the physical order and the logical (indexed) order is the same. Compare with **nonclustered index**.
code set	See **character set.**

commit	An instruction to a database to make permanent all changes made to one or more database files since the last commit or rollback operation and to make the changed records available to other users. Compare with **rollback**.
configuration file	A file that specifies the characteristics of a system or subsystem.
configuration set	A section into which service library configuration files are divided.
connection specification	Information required to make an Open ClientConnect or Open ServerConnect™ connection. The connection specification consists of the server name, platform, Net-Library™ driver name, and address information required by the Net-Library driver being used.
conversion	The transformation between values that represent the same data item but which belong to different datatypes. Information can be lost due to conversion, because accuracy of data representation varies among different datatypes.
CSP	See **catalog stored procedure**.
CT-Library	See **Client-Library**.
data definition language	A language for describing data and data relationships in a database.
database management system	A computer-based system for defining, creating, manipulating, controlling, managing, and using databases.
datatype	A keyword that identifies the characteristics of stored information on a computer.
DB-Library	A Sybase and Microsoft API that allows client applications to interact with ODS applications. See also **application program interface**.
DBMS	See **database management system**.
DDL	See **data definition language**.
destination-template transfer	A transfer method in which source data is briefly put into a template where the user can specify that some action be performed on it before execution against a target database. See also **transfer**. Compare with **bulk copy transfer**.
DirectConnect	A Sybase solution that gives client applications ODBC data access. It combines the functionality of the DirectConnect architecture with ODBC to provide dynamic SQL access to target data, as well as the ability to support stored procedures and text and image pointers.

DirectConnect for DB2 UDB	A Sybase LAN-based solution that communicates with DB2 UDB mainframe host components. It incorporates the functionality of the MDI Database Gateway and the Sybase Net-Library, and includes LU 6.2 and TCP/IP support.
DirectConnect Manager	A Sybase application for Microsoft Windows that provides remote management capabilities for DirectConnect products. These capabilities include starting, stopping, creating, and copying services.
DirectConnect server	The component that provides general management and support functions (such as log file management) to service libraries.
DirectConnect Service	A named set of properties, used with a DirectConnect Service Library, to which clients connect.
DirectConnect AccessService Library	The component that provides a set of functions within the DirectConnect server environment.
dll	See **dynamic link library**.
dynamic link library	A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a dynamic link library can be shared by several applications simultaneously.
dynamic SQL	A term pertaining to the preparation and processing of SQL source statements within a program while the program runs. The SQL source statements are contained in host-language variables rather than coded directly into the application program. Compare with **static SQL**.
ECDA for DB2 UDB	The data access product that provides access to DB2 UDB databases. Also known as DirectConnect for DB2 UDB.
ECDA for Informix	The data access product that provides access to Informix data sources. Also known as DirectConnect for Informix.
ECDA for Microsoft SQL Server	The data access product that provides access to Microsoft SQL Server data sources. Also known as DirectConnect for Microsoft SQL Server.
ECDA for Informix	The data access product that provides access to non-Sybase data sources. Also known as DirectConnect for ODBC.
embedded SQL	A SQL statement embedded within a source program and prepared before the program executes. After it is prepared, the statement itself does not change, although values of host variables specified within the statement can change.
Enterprise Connect Data Access (ECDA)	The name of the data access options that replaces the previous DirectConnect product names.

event handler	A device that processes requests and manages client communication.
global variable	System-defined variables that DirectConnect or the client application updates on an ongoing basis.
globalization	The combination of internationalization and localization. See also **internationalization**, **localization**.
interfaces file	An operating system file that must be available on each machine from which connections to DirectConnect for ODBC or other Sybase products are made. Each entry in the file determines how the host client software connects to the Sybase product.
internationalization	The process of extracting locale-specific components from the source code and moving them into one or more separate modules, making the code culturally neutral so it can be localized for a specific culture. See also **globalization**. Compare with **localization**.
keyword	A word or phrase reserved for exclusive use by Transact-SQL.
localization	The process of preparing an extracted module for a target environment, in which messages are displayed and logged in the user's language. Numbers, money, dates, and time are represented using the user's cultural convention, and documents are displayed in the user's language. See also **globalization**. Compare with **internationalization**.
MDI Database Gateway	An MDI legacy product that gives client applications access to supported data sources, such as DB2 UDB.
Net-Library	A Sybase product that lets PC applications become clients of Adaptive Server or Open Server. See also **client**, **Open Server**.
nonclustered index	An index that stores key values and pointers to data. Compare with **clustered index**.
ODBC	See **Open Database Connectivity**.
ODS	See **Open Data Services**.
OmniConnect	The functionality incorporated into ASE as CIS, referred to as ASE/CIS. See **ASE/CIS**.
Open Client	A Sybase component of SDK that provides customer applications, third-party products, and other Sybase products with the interfaces required to communicate with Open Server and Open Server applications.

Open ClientConnect	A Sybase product that provides capability for the mainframe to act as a client to LAN-based resources.
Open Data Services	A product that provides a framework for creating server applications that respond to DB-Library clients. See also **DB-Library**.
Open Database Connectivity	A Microsoft API that allows access to both relational and nonrelational databases.
Open Server	A Sybase product that provides the tools and interfaces required to create a custom server.
Open ServerConnect	A Sybase product that provides capability for programmatic access to mainframe data.
parameter	A variable with a constant value for a specified application that can denote the application. Compare with **property**.
Partner Certification Reports	Sybase publications that certify third-party or Sybase products to work with other Sybase products.
precision	The maximum number of digits that can be represented in a decimal, numeric, or float column.
precision minus scale	The number of digits to the left of the decimal point.
primary database	In transfer processing, the database accessed by the access service in a transfer statement. Compare with **secondary database**.
property	A setting for a server or service that defines characteristics, such as how events are logged or how datatypes are converted. Compare with **parameter**.
protocol	A set of rules that governs the behavior of the computers communicating on a network.
Registry	The part of the Windows NT operating system that holds configuration information for a particular machine.
relational operators	Operators supported in search conditions.
relops	See **relational operators**.
remote procedure call	A stored procedure executed on a different server from the one onto which a user is logged or on which the initiating application resides.
remote systems management	A feature that allows a System Administrator to manage multiple DirectConnect servers and multiple services from a client.

request	One or more database operations an application sends as a unit to the database. During a request, the application gives up control to the DBMS and waits for a response. See also **commit**, **rollback**, and **unit of work**.
rollback	An instruction to a database not to implement the changes requested in a unit of work and to return to the pre-transaction state. See also **transaction** and **unit of work**. Compare with **commit**.
RPC	See **remote procedure call**.
scale	The maximum number of digits that can be stored to the right of the decimal point by a numeric or decimal datatype.
secondary connection	The connection specified in the transfer statement. It represents anything that can be accessed using Open ClientConnect, such as Adaptive Server or another access service.
secondary database	In transfer processing, the supported database that is specified in the transfer statement. Compare with **primary database**.
server	A functional unit that provides shared services to clients over a network. See also **client/server**. Compare with **client**.
server process ID	A positive integer that uniquely identifies a client connection to the server.
service	A functionality available to DirectConnect applications. It is the pairing of a service library and a set of specific configuration properties.
service library	A set of configuration properties that determines service functionality. Examples of service libraries include access service libraries and administrative service libraries. See also **Access Service Library** and **Administrative Service Library**.
service name redirection	A type of service name resolution that allows a System Administrator to map alternative connections to services. See also **service name resolution**. Compare with **direct resolution**.
service name redirection file	The default name of the file used for the service name redirection feature. See also **service name redirection**.
service name resolution	The DirectConnect server mapping of an incoming service name to an actual service. See also **direct resolution**, **service name redirection**.
SNRF	See **service name redirection file**.

Software Developer's Kit (SDK)	The name of the product that contains Open Client/C, Open Client ESQL C, Open Client ESQL COBOL, language modules, jConnect for JDBC, JRE, and ODBC, OLE DB, and ADO.NET (for Windows).
SPID	See **server process ID**.
SQL	See **structured query language**.
SQL descriptor area	A set of variables used in the processing of SQL statements.
SQL stored procedure	A single SQL statement that is statically bound to the database. See also **stored procedures**.
SQLDA	See **SQL descriptor area**.
sqledit	A utility for creating and editing *sql.ini* files and file entries.
sql.ini	The interfaces file containing definitions for each DirectConnect server to which a workstation can connect. See also **interfaces file**.
statement	A single SQL operation, such as select, update, or delete.
static SQL	SQL statements that are embedded within a program and prepared before the program runs. The statement itself does not change, although values of host variables specified by the statement can change. Compare with **dynamic SQL**.
stored procedures	A collection of SQL statements and optional control-of-flow statements stored under a particular name. See also **Catalog Stored Procedure**, **SQL stored procedure**, and **system stored procedure**.
structured query language	An IBM industry-standard language for processing data in a relational database.
Sybase SQL Server	See **Adaptive Server**.
System Administrator	The user in charge of server system administration. For DirectConnect, the user responsible for installing and maintaining DirectConnect servers and DirectConnect service libraries.
system stored procedure	A Sybase-supplied stored procedure that returns information about the access service and the target database. See also **stored procedures**.
table	An array of data or a named data object that contains a specific number of unordered rows. Each item in a row can be identified unambiguously by means of one or more arguments.

Tabular Data Stream	An application-level protocol that Sybase clients and servers use to communicate.
target	A system, program, or device that interprets and replies to requests received from a source.
target database	The database to which DirectConnect transfers data or performs operations on specific data.
TDS	See **Tabular Data Stream**.
transaction	An exchange between a program on a local system and a program on a remote system that accomplishes a particular action or result.
Transact-SQL	A Sybase-enhanced version of the SQL database language used to communicate with Adaptive Server.
transfer	A DirectConnect feature that allows users to move data or copies of data from one database to another. See also **bulk copy transfer** and **destination-template transfer**.
trigger	A form of stored procedure that automatically executes when a user issues a change statement to a specified table.
T-SQL	See **Transact-SQL**.
unit of work	One or more database operations grouped under a commit or rollback. A unit of work ends when an application commits or rolls back a series of requests, or when the application terminates. See also **commit**, **rollback**, and **transaction**.
view	An alternative representation of data from one or more tables. A view can include all or some of the columns contained in the table or tables on which it is defined.
wildcard	A special character that represents a range of characters in a search pattern.

Index