SYBASE®

User’s Guide for DB2 Access Services

Mainframe Connect DirectConnect™ for z/OS Option

12.6

[Microsoft Windows and UNIX]

DOCUMENT ID: DC36455-01-1260-01

LAST REVISED: February 2005

Contents

About This Book

This guide describes how to configure and use a DirectConnect™ for z/OS access service, including information about access service configuration properties, datatype conversion, request processing, data transfer, and stored procedures.

This guide also provides information about compatibility with MDI Database Gateway™.

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 also provide product administration, troubleshooting, and disaster recovery.

How to use this book

The following table shows how this book is organized.

See	To
Chapter 1, “Introducing DirectConnect for z/OS”	Learn about DirectConnect for OS/390 and the access services.
Chapter 2, “Creating and Configuring DB2 Access Services”	Configure the Access Service Library and DB2 access service properties.
Chapter 3, “Querying and Setting Operating Values”	Use global variables and set statements to query and set operating values for your client connections.
Chapter 4, “Converting Datatypes”	Review datatype conversions between DB2 UDB and Open Client™ and Open Server™.
Chapter 5, “Understanding the Request Process”	Begin the SQL request process.

See	To
Chapter 6, “Issuing SQL Statements”	Review SQL transformation modes and cursor, dynamic, and language commands.
Chapter 7, “Issuing Remote Procedure Calls”	Issue SQL Server® remote procedure calls (RPCs) that initiate events in remote databases.
Chapter 8, “Understanding the Transfer Process”	Review the transfer process, including transfer targets, datatype conversion for transfer processing, errors and error handling, and backward compatibility.
Chapter 9, “Using Bulk Copy Transfer”	Use bulk copy transfer to transfer data.
Chapter 10, “Using Destination-Template Transfer”	Use destination-template transfer to transfer data.
Chapter 11, “Accessing Catalog Information with CSPs”	Obtain information about database objects that you need to access the DB2 UDB database catalog. Catalog stored procedures (CSPs) provide this catalog access.
Chapter 12, “Retrieving Information with System Procedures”	Use system procedures.
Chapter 13, “RSPs and Host-Resident Requests”	Create, execute, and delete remote stored procedures (RSPs) and host-resident requests.
Appendix A, “Configuration References and Code Set Tables”	Access a synopsis of configuration properties and code set identifiers, and how they interact in transaction management.
Appendix B, “Compatibility with MDI Database Gateways and Net-Gateway”	Identify information regarding the compatibility between DirectConnect and both the MDI Database Gateway and Net-Gateway™.
Appendix C, “Using Sybase Mode Commands.”	Issue structured query language (SQL) commands that an access service recognizes and acts upon when the SQL transformation mode is set to sybase mode.

Related documents

To install DirectConnect products and DirectConnect Manager, use the Mainframe Connect DirectConnect for z/OS Option *Installation Guide*.

To configure and administer the DirectConnect server, use the ECDA Mainframe Connect *Server Administration Guide* for DirectConnect.

To configure and administer the DirectConnect Transaction Router Service Library (TRS) and to administer Mainframe Client Connect, use the Mainframe Connect DirectConnect for z/OS Option *User's Guide for Transaction Router Services*.

To install and administer the other Mainframe Connect Integrated Product Set (IPS) products, use the following documents:

For additional references, use the following documents:

- Mainframe Connect Client Option *Installation and Administration Guide* for CICS
- Mainframe Connect Client Option *Installation and Administration Guide* for IMS and MVS
- Mainframe Connect Client Options *Programmer's Reference* for Remote Stored Procedures
- Mainframe Connect Server Option *Installation and Administration Guide* for CICS
- Mainframe Connect Client Option *Installation and Administration Guide* for IMS and MVS
- Mainframe Connect Client Options *Programmer's Reference* for Client Services Applications
- Open Client and Open Server *Common Libraries Reference Manual*
- Open Client *Client-Library/C Programmer's Guide*
- Open Client *Client-Library/C Reference Manual*
- Open Server *Server-Library/C Reference Manual*

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:

This type of information	Looks like this
Gateway-Library function names	TDINIT, TDCANCEL
Client–Library™ function names	CTBINIT, CTBCANCEL
Other executables (DB-Library™ routines, SQL commands) in text	the dbrpcparam routine, a select statement
Directory names, path names, and file names	*/usr/bin directory, interfaces* file
Variables	*n* bytes
SQL Server datatypes	datetime, float
Sample code	`01 BUFFER PIC S9(9) COMP SYNC`
User input	*01 BUFFER PIC X(n)*
Client-Library and Gateway-Library function argument names	*BUFFER, RETCODE*
Names of objects stored on the mainframe	SYCTSAA5
Symbolic values used with function arguments, properties, and structure fields	CS_UNUSED, FMT_NAME, CS_SV_FATAL
Client-Library property names	CS_PASSWORD, CS_USERNAME
Client-Library and Gateway-Library datatypes	CS_CHAR, TDSCHAR

All other names and terms are in regular typeface.

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 guide.

Symbol	Convention
( )	Parentheses are 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 for z/OS

This chapter provides an overview of DirectConnect for z/OS. DirectConnect for z/OS when combined with other Sybase products, provides access and integration of mainframe data. For more information about other Sybase products, see the *Overview Guide* for Mainframe Connect.

This chapter contains the following topics:

Topic	Page
DirectConnect for z/OS overview	1
DirectConnect for z/OS environment	8

DirectConnect for z/OS overview

This section describes DirectConnect for z/OS and other Sybase products that DirectConnect interacts with. This section covers the following topics:

- DirectConnect architecture
- DirectConnect components
- Related products

DirectConnect architecture

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

DirectConnect serves as a fundamental building block for highly-scalable database middleware applications. DirectConnect products are local area network (LAN)-based middleware gateways and 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 Integration Services (ASE/CIS), Sybase Adaptive Server, and Replication Server®.

DirectConnect for z/OS consists of:

- A server, which provides the framework in which service libraries can operate
- One or more service libraries (DB2 and TRS), which provide the framework in which access services can operate
- One or more access services for each service library (DB2 and TRS), which are the logical points of connection for DirectConnect clients.

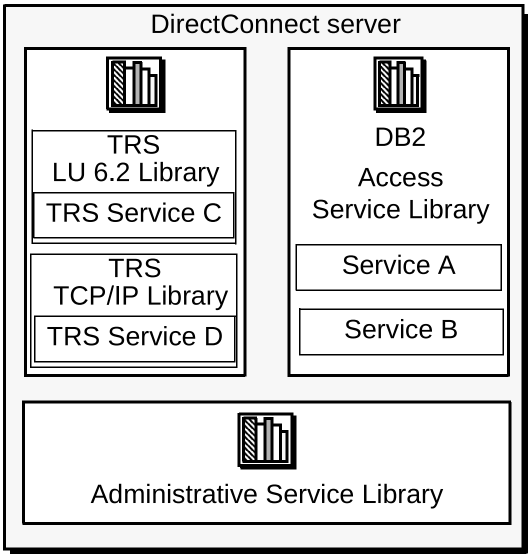
The following subsections describe each of these components.

DirectConnect components

This section describes the following DirectConnect components:

- DirectConnect server
- DirectConnect service libraries
- DIrectConnect access services

Figure 1-1: DirectConnect server, service libraries, and 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 for access services. TRS has its own Tabular Data Stream ™(TDS) trace file, LU 6.2 protocol trace file, and TCP/IP protocol trace file.
- 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

Residing on the DirectConnect server, a service library is a set of configuration properties that describes how its access services will function. The following service libraries reside on the DirectConnect server:

- DB2 Access Service Library
- Transaction Router Service Library
- Administrative Service Library

DIrectConnect access services

An access service is the client connection point for a DirectConnect server. It is the pairing of a service library with a set of specific values for the configuration properties.

DB2 access services

A DB2 access service works with MainframeConnect™ DB2 UDB Option to allow clients to access DB2 data in a DB2 UDB database.

Each DB2 access service is a specific set of configuration properties that perform the following:

- Transform SQL
- Convert datatypes
- Support RPCs
- Transfer data between DB2 UDB and other servers accessible through Open Client
- Support catalog stored procedures (CSPs) and system stored procedures
- Support RSPs and host-resident requests

Transaction Router Service (TRS)

Each TRS library contains a TRS that provides access to DB2 data and supports Open ServerConnect™ mainframe applications, defined to TRS as remote procedure calls (RPCs).

TRS routes requests from remote LAN-based clients to Open ServerConnect transactions. Optionally, it can also route requests to MainframeConnect DB2 UDB Option and return results to the client.

Having multiple instances of a TRS library on a server results in different physical copies of the dynamic link library or shared library files that constitute the TRS component.

Security can also be configured on a transaction or user basis.

There are two TRS libraries:

- *TRSLU62* service library, which uses the LU 6.2 communications protocol to talk to Mainframe Connect or to any Open ServerConnect application running in CICS.
- *TRSTCP* service library, which uses the Transmission Control Protocol/Internet Protocol (TCP/IP) communications protocol to talk to MainframeConnect or any Open ServerConnect application running in CICS.

Having multiple instances of a TRS library on a server results in different physical copies of the shared library files that constitute the TRS component.

For an explanation of the TRS components of DirectConnect, see the DirectConnect *Transaction Router Service User's Guide*.

Administrative service library

The Administrative service library provides specific administrative services for all DirectConnect libraries, including logging, tracing, and allowing remote configuration of DirectConnect access services (for example, through DirectConnect Manager).

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.

Related products

This section describes products that DirectConnect interacts with to provide mainframe access for LAN client requests.

MainframeConnect DB2 UDB Option

MainframeConnect DB2 UDB Option is a CICS transaction that works with DirectConnect for z/OS to provide access to mainframe data. It performs the following functions:

- Supports full read-write, dynamic SQL access to data
- Allows applications to use cursors for flexible and efficient result set processing
- Permits the use of long-running transactions against mainframe databases
- Allows applications to use dynamic events to map SQL to a static plan

DirectConnect for z/OS invokes Mainframe Connect for DB2 UDB to access mainframe data on behalf of its Open Client-based clients, such as:

- ASE/CIS
- Adaptive Server through remote procedure calls (RPCs)
- Enterprise Application Server
- JDBC or ODBC applications
- Replication Server

Note MainframeConnect for DB2 UDB is available only for MVS-CICS environments.

Open ServerConnect

Open ServerConnect is a programming environment that lets you create mainframe transactions that Sybase client applications can access. To provide this access, Open ServerConnect uses the following basic interfaces:

- Traditional Open Server programming environment (for new customers and Sybase-heritage customers using new applications)
- remote stored procedure (RSP) programming environment (only for MDI-heritage customers)

These mainframe transactions provide access to virtually any z/OS data source and are used for a variety of functions, including:

- Accessing existing mainframe applications
- Initiating mainframe batch jobs
- Providing source data for data transfer operations
- Providing data mapped to a table within the CIS functionality in ASE/CIS (formerly OmniConnect), thus allowing results to be accessed or joined with data from other targets

LAN-side client applications access Open ServerConnect transactions directly through DirectConnect or indirectly through ASE/CIS, or through a Sybase Adaptive Server RPC.

Open ClientConnect

Open ClientConnect™ is a programming environment that lets you create mainframe applications that access:

- LAN data residing on an Adaptive Server or other supported data sources
- Other CICS regions
- Mainframe Client Connect

It allows you to treat the mainframe as just another node on a LAN.

Open ClientConnect uses the following APIs:

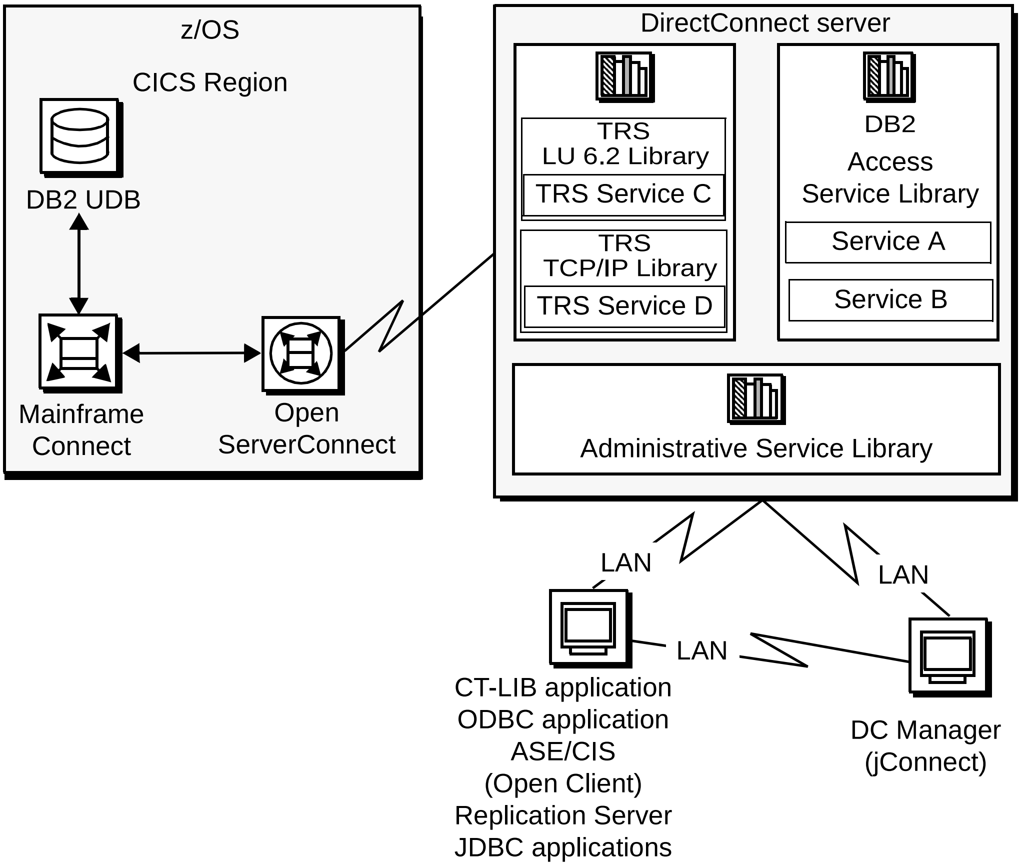
- Traditional Open Client programming environment (for new customers and Sybase-heritage customers using new applications)
- Client Services Application (CSA) programming environment (only for MDI-heritage customers)

DirectConnect for z/OS environment

DirectConnect for z/OS consists of a server and one or more service libraries. The server provides the framework in which the service libraries operate. The DB2 Access Service Library is the program component of DirectConnect for z/OS that works with MainframeConnect for DB2 UDB to access DB2 data.

The following figure shows the relationship of the DB2 Access Service Library with components of the client workstation, LAN, and mainframe environments.

Figure 1-2: DirectConnect for z/OS environment



As shown, the request from a client application uses the LAN to communicate with the DirectConnect server. From there, either TRS or a DB2 access service routes the request to the appropriate CICS region. Then the request accesses data on the DB2 UDB database.

For more information on how to create multiple TRS libraries, see the Mainframe Connect DirectConnect *Transaction Router User's Guide*.

CHAPTER 2

Creating and Configuring DB2 Access Services

This chapter describes how to configure properties to customize the DB2 Access Service Library and individual DB2 access services.

This chapter contains the following topics:

Note Appendix A contains a quick reference for the configuration properties.

General information

Note For this document, all references to the DB2 access service will be referred to simply as the DB2 access service.

You change existing properties and create DB2 access services in either of the following ways:

- Using DirectConnect Manager

  If you use DirectConnect Manager, which is supported on Windows and UNIX platforms, you can configure DB2 access service properties and dynamically change many properties without stopping and starting the server. DirectConnect Manager is easier to use, provides you with a GUI interface, and performs error checking. For information about using DirectConnect Manager, see the DirectConnect Manager online help.

- Manually editing the DB2 Access Service Library configuration file that resides on the DirectConnect server

  If you edit the configuration file, you must stop and restart the server for the changes to take effect. The instructions in this chapter describe how to edit the DB2 Access Service Library configuration file.

The DB2 Access Service Library uses some configuration information from the server. For instructions about configuring DirectConnect server properties, see the DirectConnect *Server Administration Guide*.

Configuration file

Note Sybase recommends that you use DirectConnect Manager to configure the DB2 Access Service Library, modify configuration property values, and create DB2 access services. Using DirectConnect Manager allows you to make changes dynamically without restarting the server.

To configure the DB2 Access Service Library and create and modify DB2 access services, you must use the DB2 Access Service Library configuration file. It is a simple text file named *db2.cfg*. Use your text editor to change any DB2 access service property, then save the configuration file. To find the location of this file within the DirectConnect directory structure, see the Mainframe Connect DirectConnect for z/OS Option *Installation Guide* for your platform.

Each DB2 access service has a specific set of configuration properties. To configure a DB2 access service, enter site-specific values for all required properties. When you make changes to DB2 access service properties that are not required, enter only the values that differ from the default values.

Configuration file format

A service library configuration file consists of:

- A primary section [*Service Library*] that groups DB2 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. Each comment must be on a separate line and begin with a semicolon (;) or crosshatch (#) in column one.

Sample configuration file

Following is an example of two DB2 Access Service Library configuration files:

```
[Service Library]
{Logging}
LogSvclibStatistics=60

#The configuration for the first service
[zOS_sna]

{ACS Required}
DefaultClientCodeset=OpenServerDefault
```

```
DefaultTargetCodeset=OpenServerDefault
UseClientCharset=yes
TPName=AMD2
ConnectionProtocol=lu62
ConnectionSpec1=LOCAL
ConnectionSpec2=CICSAMD2
ConnectionSpec3=MVSMODE
{Client Interaction}
EnableAtStartup=yes
TransactionMode=short
GatewayCompatible=no
{Target Interaction}
Allocate=connect
SQLTransformation=sybase
QuotedStringDelimiter='
{Data Conversion Errors}
DateTimeConvertError=reject
{Datatype Conversion}
DateTimeResults=datetime
{Logging}
LogTargetActivity=yes
LogReceivedSQL=yes
LogTransformedSQL=yes
{Tracing}
TraceTarget=no

#The configuration for the second service
[zOS_tcp]

{ACS Required}
DefaultClientCodeset=OpenServerDefault
DefaultTargetCodeset=OpenServerDefault
UseClientCharset=yes
TPName=AMD2
ConnectionProtocol=tcpip
ConnectionSpec1=ophelia
ConnectionSpec2=7021
ConnectionSpec3=CICSAMD2
{Client Interaction}
EnableAtStartup=yes
TransactionMode=short
GatewayCompatible=no
{Target Interaction}
Allocate=connect
SQLTransformation=sybase
QuotedStringDelimiter='
{Data Conversion Errors}
```

```
DateTimeConvertError=reject
{Datatype Conversion}
DateTimeResults=datetime
{Logging}
LogTargetActivity=yes
LogReceivedSQL=yes
LogTransformedSQL=yes
{Tracing}

TraceTarget=no
```

DB2 access service names

As shown in the previous example, access service names must conform to the following rules:

- Access service names must be unique (without regard to case).
- Access service names must not exceed 31 characters in length 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, with *no* spaces.

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, go to the Managing Access Service 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 the DirectConnect Manager online Help topic, “Connecting DirectConnect Manager to a DirectConnect Server.”

For additional information, refer to the 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 DB2 Access Service Library configuration file. Update the service library configuration properties.
2. Open the DB2 access services file and change DB2 access service property values as applicable.
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 DB2 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 DB2 Access Service Library configuration file called *db2.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 DB2 access service from a client machine, you must enter the DB2 access service name in the *sql.ini* configuration file on Windows NT machines or the interfaces 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 Mainframe Connect DirectConnect for z/OS Option *Installation Guide* for your platform.

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

Configuration properties

The following principles apply to DB2 Access Service Library and DB2 access service properties:

- All properties are grouped into categories.

- Service library properties apply to the DB2 Access Service Library as a whole.
- DB2 access service properties apply to specific DB2 access services.
- Configuration properties are not case sensitive.

Property categories

The following sections describe the property categories:

- ACS Required properties
- Catalog Stored Procedure properties
- Client Interaction properties
- Data Conversion Error properties
- Datatype Conversion properties
- Logging properties
- Target Interaction properties
- Tracing properties
- Transfer properties

See Appendix A, “Configuration References and Code Set Tables,” for an alphabetized listing of all configuration properties within the DB2 Access Service Library.

ACS Required properties

Some DB2 access service properties require specific values for your site. These required properties reside in the ACS Required property category. Be sure to supply these values following your installation.

The subsection heading and properties must appear in the service library configuration file as follows:

{ACS Required}

ConnectionProtocol
ConnectionSpec1
ConnectionSpec2
ConnectionSpec3
DefaultClientCodeset (see code set translation)
DefaultTargetCodeset (see code set translation)
UseClientCharset (see code set translation)
TPName

Each of the required properties is described in the following subsections.

The following code set translation section describes the relationship between three of the ACS required properties; DefaultClientCodeset, DefaultTargetCodeset, and UseClientCharset.

Code set translation

To determine if code set translation is performed by the DirectConnect access service or by the target host, the values of the three following configuration properties must be considered. Each configuration property is described in the following sections:

- DefaultClientCodeset
- DefaultTargetCodeset
- UseClientCharset

Recommended default values

The recommended default values for the three configuration properties are:

```
DefaultClientCodeset = OpenServerDefault
Default TargetCodeset = OpenServerDefault
UseClientCharset = yes
```

Using the recommended default values causes *no translation* to take place by the DirectConnect access service and results in translation at the target, using the client code set value in the client *login* record.

The list of certified code set identifiers are available in the *Unilib Reference Manual Developer's Kit for Unicode*.

Note OpenServerDefault is the default and becomes the value assigned, by platform, in the server's *locales.dat* file.

Configuration property relationships

Code set translation takes place either in the DirectConnect access service or at the target host based on the following relationships:

- If the values of the DefaultClientCodeset and the DefaultTargetCodeset are equal, and the UseClientCharset is equal to *yes*, then translation takes place at the target, using the value from the client *login* record.
- If the values of the DefaultClientCodeset and the DefaultTargetCodeset are equal, and the UseClientCharset is equal to *no*, then translation takes place at the target host using the DefaultClientCodeset value.
- If the values of the DefaultClientCodeset and the DefaultTargetCodeset are *not* equal, and the UseClientCharset is equal to *yes*, then translation takes place in DC using the value from the client *login* record.
- If the values of the DefaultClientCodeset and the DefaultTargetCodeset are *not* equal, and the UseClientCharset is equal to *no*, then translation takes place at the target host using the DefaultClientCodeset value.

ConnectionProtocol

Specifies the communication protocol the DB2 access service uses to connect to the target database.

Syntax	ConnectionProtocol=[lu62 | tcpip]
Default	lu62
Values	lu62 specifies LU 6.2 as the communications protocol that the DB2 access service uses to connect to the target database. tcpip specifies TCP/IP as the communications protocol that the DB2 access service uses to connect to the target database.

ConnectionSpec1

Specifies the platform-specific LU 6.2 connection information or the IP address for TCP/IP communications.

Syntax	ConnectionSpec1=*char*
Range	1–255 characters
Default	No default
Values	• For TCP/IP, the *name* or *IP address* of the host.

- For LU 6.2 communications, the value is platform-dependent:
  - Windows NT: *Local LU alias*
  - Solaris: *Local LU alias*
  - HP: *Local LU alias*
  - AIX: *Local LU alias*

Comments

The communications protocol determines the maximum acceptable length for this property value.

ConnectionSpec2

Specifies the platform-specific LU 6.2 connection information or the port number for TCP/IP communications.

Syntax

ConnectionSpec2=*char*

Range

1–255 characters

Default

No default

Values

- For TCP/IP, the *port number* that the CICS region is listening on.
- For LU 6.2 communications:
  - Windows NT: *Remote LU Alias*
  - Solaris: *Remote LU Alias*
  - HP: *Remote LU Alias*
  - AIX: Remote LU Alias

Comments

The communications protocol determines the maximum acceptable length for this property value.

ConnectionSpec3

Specifies the SNA mode name for LU 6.2 communications or the CICS region name running Open ServerConnect for TCP/IP communications.

Syntax

ConnectionSpec3=*char*

Range

1–255 characters

Default

No default

Values

- For SNA, the SNA mode name for LU 6.2 communications
- For TCP/IP, the CICS region name running Open ServerConnect for TCP/IP communications

Comments

The communications protocol determines the maximum acceptable length for this property value.

DefaultClientCodeset

Specifies the code set the client application uses when sending information to the DB2 access service. For an explanation of the relationship with the DefaultTargetCodeset and the UseClientCharset, see "Code set translation" on page 19.

Syntax

DefaultClientCodeset=[clientcodeset | OpenServerDefault]

Range

1–255 characters

Default

OpenServerDefault

Values

To operate correctly, the DB2 access service requires a DefaultClientCodeset value that accurately reflects the code set of the client application.

Enter one of the following as the DefaultClientCodeset value:

- The client application code set identifier.
- The default setting, OpenServerDefault, becomes the value assigned by platform in the DirectConnect server's *locales.dat* file.

Comments

- For information regarding the relationship of the DefaultClientCodeset, DefaultTargetCodeset, and the UseClientCharset refer to the section titled, "Code set translation" on page 19.
- When data passes from the client application to the target database, the DB2 access service converts the data from the DefaultClientCodeset to the DefaultTargetCodeset if the two property values are unequal.
- The DB2 access service obtains the client application language from one of the following:
  - The login record
  - The server DefaultServerLanguage configuration property value

For information about the DefaultServerLanguage property that defines the language in which the client messages that originate in the DirectConnect server are returned, see the DirectConnect *Server Administration Guide*.

DefaultTargetCodeset

Specifies the target code set the DB2 access service uses to convert characters when it sends data to the target database. For an explanation of the relationship with the DefaultClientCodeset and the UseClientCharset, see “Code set translation” on page 19.

Syntax	DefaultTargetCodeset=[targetcodeset | OpenServerDefault]
Range	1–255 characters
Default	OpenServerDefault
Values	The DefaultTargetCodeset value name corresponds to a code set identifier.

Comments

- For information regarding the relationship of the DefaultClientCodeset, DefaultTargetCodeset, and the UseClientCharset, refer to the “Code set translation” on page 19.
- When data passes from the target database to the client application, the DB2 access service converts the data from the DefaultTargetCodeset to the DefaultClientCodeset. (See “DefaultClientCodeset” on page 22.)
- The DefaultTargetCodeset property enables the DB2 access service to support single-byte or mixed-byte language machines.

UseClientCharset

Specifies how the UseClientCharset is used in the data translation process. For an explanation of the relationship with the DefaultClientCodeset and the DefaultTargetCodeset, see the section, “Code set translation” on page 19.

Syntax	UseClientCharset= [yes | no]
Default	yes
Value	yes, no

Comments

yes causes DirectConnect to synchronize the DefaultClientCodeset and the client’s login charset, thus using the client’s charset for all translations.

no causes DirectConnect to have translation take place at the target host, using the DefaultClientCodeset value.

TPName

The CICS transaction program (TP) name for a specific MainframeConnect for DB2 program. The DB2 access service accesses this TP name on the mainframe.

Syntax	TPName= [tpname | AMD2]
Range	1–8 characters
Default	AMD2
Values	tpname is the CICS transaction program (TP) name for a specific MainframeConnect for DB2 program.

Catalog Stored Procedure properties

These properties control the information a DB2 access service returns from catalog stored procedures (CSPs). For more information about CSPs, see Chapter 11, “Accessing Catalog Information with CSPs.”

The subsection heading and the properties must appear in the service library configuration file as:

{Catalog Stored Procedures}

CSPCatalogQualifier
CSPDBName
CSPExclusions
CSPIncludeAlias
CSPIncludeSynonym
CSPIncludeSystem
CSPIncludeTable
CSPIncludeView
CSPQualByDBName
DatatypeInfo
SQLInformationFile

CSPCatalogQualifier

Specifies the qualifier of the system tables a DB2 access service uses when processing CSPs.

Syntax	CSPCatalogQualifier=[*qualifier* | SYSIBM]
Range	0–255 characters
Default	SYSIBM
Values	*qualifier* is the name of the system catalog.
	If you do not specify a value, the default is SYSIBM.

CSPDBName

Specifies the name of the database about which a DB2 access service returns information from the sp_tables CSP.

Syntax	CSPDBName=*dbname*
Range	0–255 characters
Default	No default
Values	*dbname* is a database name from the DB2 system catalog.

CSPExclusions

Specifies whether a DB2 access service limits access to information normally returned from the sp_tables CSP based upon authorization to information.

Syntax	CSPExclusions=[user | none | nonauth | nonauthpublic]
Default	user
Values	none specifies that objects are not excluded from the result set based on authorization or ownership.
	user specifies that objects are excluded from the result set based on specific user authorization information.
	nonauth specifies that non-authorized and public objects are excluded from the result set.
	nonauthpublic specifies that non-authorized objects are excluded from the result set, and public objects are included in the result set.

CSPIncludeAlias

Specifies whether the DB2 access service returns information about aliases from the sp_tables CSP.

Syntax	CSPIncludeAlias=[no | yes]
Default	no
Values	no specifies that the DB2 access service does not return information about aliases. yes specifies that the DB2 access service returns information about aliases.

CSPIncludeSynonym

Specifies whether the DB2 access service returns information about synonyms from the sp_tables CSP.

Syntax	CSPIncludeSynonym=[no | yes]
Default	no
Values	no specifies that the DB2 access service does not return information about synonyms. yes specifies that the DB2 access service returns information about synonyms.

CSPIncludeSystem

Specifies whether the DB2 access service returns information about the system tables from the sp_tables CSP.

Syntax	CSPIncludeSystem=[no | yes]
Default	no
Values	no specifies that the DB2 access service does not return information about system tables. yes specifies that the DB2 access service returns information about system tables.
Comments	The user issuing sp_tables must be authorized to query these tables.

CSPIncludeTable

Specifies whether the DB2 access service returns information about tables from the sp_tables CSP.

Syntax	CSPIncludeTable=[yes | no]
Default	yes
Values	yes specifies that the DB2 access service returns information about tables.
	no specifies that the DB2 access service does not return information about tables.

CSPIncludeView

Specifies whether the DB2 access service returns information about views from the sp_tables CSP.

Syntax	CSPIncludeView=[yes | no]
Default	yes
Values	yes specifies that the DB2 access service returns information about views.
	no specifies that the DB2 access service does not return information about views.

CSPQualByDBName

Specifies whether the value in the CSPDBName property should be used to further qualify the result set from the sp_tables CSP.

Syntax	CSPQualByDBName=[no | yes]
Default	no
Values	no specifies that the DB2 access service does not limit the result set from sp_tables to a particular database.
	yes specifies that the DB2 access service returns information from sp_tables for a particular database.
Comments	For more information, see "CSPDBName" on page 25.

DatatypeInfo

Specifies the type of datatype information returned from the sp_datatype_info CSP.

Syntax	DatatypeInfo=[transact | target]
Default	transact
Values	transact specifies that sp_datatype_info to return the Transact-SQL (T-SQL) datatypes that map to the ODBC datatypes that the target supports. target specifies that sp_datatype_info to return target datatype information.

SQLInformationFile

Specifies the file that the sp_sqlgetinfo system procedure uses for target database information.

Syntax	SQLInformationFile=[filename | db2.sif]
Range	0–128 characters
Default	db2.sif
Values	filename is the file with sp_sqlgetinfo information.

Comments

- During installation, DirectConnect places a default copy of the SQL information file for each DB2 Access Service Library into the *DC-12_6\servername\cfg* directory, where *servername* is the name of the DirectConnect server. If this property is left blank or not changed, the default file is used.
- To customize the file, you first make a copy of the default file, save it under a new file name, and then edit the custom file by following these rules:
  - Do not modify the header section of the file.
  - Begin comments with a semicolon (;) or crosshatch (#) in column 1, on a line by themselves.
  - Add additional properties to the [Start SQLGetInfo] section of the file in the property=*value* format as follows:

    ```
    SQL_ACTIVE_CONNECTIONS=0
    ```
  - Save the file.

- To use the custom file, you must enter the custom file name as the SQLInformationFile property value.
- If you move the custom file to a directory other than the *DC-12_6\servername\cfg* directory and want to use the file, enter the full path and custom file name as the SQLInformationFile property value.

Client Interaction properties

These properties control how a DB2 access service interacts with client applications.

The subsection heading for these properties must appear in the service library configuration file as:

{Client Interaction}

ApplicationValidationFile
ClientDecimalSeparator
ClientIdleTimeout
EnableAtStartup
GatewayCompatible
MaxResultSize
MaxRowsReturned
MaxSvcConnections
quoted_identifier
SendWarningMessages
ServiceDescription
StripBinaryZero
TextSize
TransactionMode
Version

ApplicationValidationFile

Points to a file with application validation information. This property is included for backward compatibility with MDI Database Gateway only. To use this feature, set the GatewayCompatible property to yes.

Syntax ApplicationValidationFile=*pathfilename*

Range	0–128 characters
Default	No default
Values	*pathfilename* is the complete file specification, including the full path name, for the file with application validation information.
Comments	• Sybase recommends that you use the service name redirection capability as described in the DirectConnect *Server Administration Guide* instead of application validation. • If you choose to use both service name redirection and application validation, service name redirection takes precedence.

ClientDecimalSeparator

Specifies the character the client application uses to separate decimal numbers for presentation purposes. The target database does not store the client decimal delimiter character.

Syntax	ClientDecimalSeparator=[. | ,]
Default	. (period)
Values	A period (.) indicates that the client application uses a period as the decimal delimiter. A comma (,) indicates that the client application uses a comma as the decimal delimiter.
Comments	• If some of your client applications use a different character as a decimal delimiter, make sure that those applications connect to a DB2 access service configuration set that uses the same client decimal delimiter character. • If the client decimal delimiter is a comma (,), set the DecimalResults property value to char to enable the DB2 access service to return decimal results with a comma as the client decimal delimiter. • If the SQLTransformation property value is passthrough, use the TargetDecimalSeparator value for all SQL requests going to the target database. For example, if the TargetDecimalSeparator value is a period (.), use a period for the decimal delimiter in all SQL requests going to the target database (such as insert statements). • For information about SQL transformation modes, see “SQL transformation modes” on page 107.

ClientIdleTimeout

Specifies how many minutes a client connection can remain inactive before a DB2 access service terminates the connection.

Syntax	ClientIdleTimeout=*integer*
Range	0–1024
Default	0
Values	*integer* is how many minutes a client connection can remain inactive before a DB2 access service terminates the connection. *0* indicates that a DB2 access service never terminates an idle connection.
Comments	• A connection is idle when: • A client connected but did not issue a command. • A command processed, but the client did not issue another command. • A large result set returned from SQL request processing, and the result window paused for the specified timeout period. • The DB2 access service checks client activity each minute. Therefore, a client can remain inactive for up to one minute beyond the ClientIdleTimeout value before the DB2 access service terminates the connection.

EnableAtStartup

Specifies whether this DB2 access service starts when the DirectConnect server starts.

Syntax	EnableAtStartup=[no | yes]
Default	no
Values	no means that the DB2 access service does not start when the server starts. yes means that the DB2 access service does start when the server starts.
Comments	If you are not using DirectConnect Manager to manage your DB2 access services, set this property to yes.

GatewayCompatible

Specifies whether the DB2 access service accepts and returns MDI Database Gateway set statements, global variables, and a few error messages that client applications use to make processing decisions. This property is included for backward compatibility with MDI Database Gateway only. Sybase recommends that you use current DB2 access service settings.

Syntax

GatewayCompatible=[no | yes]

Default

no

Values

no means that the DB2 access service recognizes DB2 access service settings only.

yes means that the DB2 access service recognizes both DB2 access service settings and MDI Database Gateway settings.

Comments

- Backward compatibility with the MDI Database Gateway is provided only as an interim measure to allow you to upgrade your client applications to the syntax supported in the DB2 access service.
- If your client applications do not require MDI Database Gateway settings, set this value to no to use DB2 access service settings only.
- If your current client applications use previous MDI Database Gateway global variables and set statements, set this property to yes to allow your applications to continue using MDI Database Gateway settings.

MaxResultSize

Specifies the maximum number of bytes a DB2 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 (zero) defaults to 2,147,483,646 bytes (the default value).

Comments

- The MaxResultSize value is approximate in that a DB2 access service checks at the end of each row to see if the MaxResultSize value was exceeded.
- If the MaxResultSize value is exceeded, the DB2 access service:

    - Sends the entire row to the client application (not a partial row).
    - Does not send any of the remaining rows in the result set.
    - Issues a warning message.
- Typically, the number of bytes returned is less than the value of the MaxResultSize property because the calculations for character columns are based on the defined size, whereas the actual data contained in these columns is usually less.
- The SendWarningMessages property controls whether the DB2 access service returns warning messages to the client application. See “SendWarningMessages” on page 35.

MaxRowsReturned

Specifies the maximum number of rows a DB2 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 (zero) defaults to 2,147,483,646 rows (the default value).

Comments

- If the MaxRowsReturned value is exceeded, the DB2 access service issues a warning message and does not send any of the remaining rows in the result set.
- The SendWarningMessages property controls whether the DB2 access service returns warning messages to the client application. See “SendWarningMessages” on page 35.

MaxSvcConnections

Specifies the maximum number of client connections the DB2 access service can handle at one time.

Syntax	MaxSvcConnections=[*integer* | MaxConnections]

Range — 1–*n*, where *n* is the maximum number of client connections allowed by the server.

Default — MaxConnections property value of the DirectConnect server

Values — *integer* is a number of client connections.

Comments

- The server MaxConnections property determines the maximum number of client connections. For information about MaxConnections, see the DirectConnect *Server Administration Guide*.
- The Allocate property value can affect the number of client connections:
    - If the Allocate property is set to connect, MaxSvcConnections should be equal to or less than the number of parallel APPC sessions that can be supported from the server to DB2.
    - If the Allocate property is set to request, you can set the MaxSvcConnections property to a value greater than the number of parallel APPC sessions that you configured in your APPC software. For more information about the Allocate property, see "Allocate" on page 57.

    Note The DB2 access service does not verify the validity of the MaxSvcConnections value.

- For information about APPC connections, see your APPC software documentation for your specific DBMS and platform.

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 DB2 to treat them as identifiers. The following example shows the correct way to insert a character string into col1 of one table 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 a DB2 access service returns warning messages to the client application.

Syntax	SendWarningMessages=[no | yes]
Default	no
Values	no specifies the DB2 access service not to return warning messages to the client application.

	yes specifies the DB2 access service to return warning messages to the client application.
Comments	For information about error messages, see the DirectConnect *Error Message Guide*.

ServiceDescription

	Describes a DB2 access service.
Syntax	ServiceDescription=*char*
Range	0–255 characters
Default	No default
Values	*char* is a user-defined character string.
Comments	This property allows you to place descriptive information in the configuration file about each DB2 access service.

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.

TextSize

	Specifies the maximum number of bytes in character columns a DB2 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 (zero) defaults to *2147483647* (the default value).
Comments	• A DB2 access service truncates data exceeding the TextSize length and does not issue a warning message. • The DB2 access service returns character data longer than the value of the XNLChar and XNLVarChar configuration property values as CS_TEXT datatype. • For DirectConnect for DB2 access service, 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 2147483647 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.

TransactionMode

Specifies whether the DB2 access service or the client application manages commits and rollbacks.

Syntax	TransactionMode=[short | long]
Default	short
Values	long specifies the DB2 access service to give commitment control to the client application. The DB2 access service holds open the connection to the target database until the client application issues a commit or rollback or until the ClientIdleTimeout value (see “ClientIdleTimeout” on page 31) is exceeded. If ClientIdleTimeout is exceeded, the transaction rolls back. short indicates that the DB2 access service issues a commit or a rollback after each request.

Version

Allows you to customize an alternate version string for client applications that rely on a particular string that is different from the DB2 access service default version string.

Syntax	Version=*versionstring*
Default	The DB2 access service default version string
Values	*versionstring* is the version string you want reported to client applications.

Comments

- If you leave this property blank, it defaults to the DB2 access service default version string.
- If you customize an alternate version string, the following rules apply:
  - The format of this string cannot contain embedded new lines.
  - You can insert a space after the equal sign (=) for readability in the configuration file; however, when a DB2 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 211).

Data Conversion Error properties

These properties control the action a DB2 access service takes when it encounters data conversion errors.

The subsection heading for these properties must appear in the service library configuration file as:

{Data Conversion Errors)

CharConvertError
DateTimeConvertError
DefaultDate
DefaultNum
DefaultTime
NumConvertError

CharConvertError

	Specifies the action a DB2 access service takes when it encounters a result column that is too long for the target column.
Syntax	CharConvertError=[reject | truncate]
Default	reject
Values	reject specifies the DB2 access service to reject the row containing the error and issue a warning message.
	truncate specifies the DB2 access service to insert data to the length of the target column, truncate the remaining data, and issue a warning message.
Comments	The SendWarningMessages property controls whether the DB2 access service returns warning messages to the client application. (See "SendWarningMessages" on page 35.)

DateTimeConvertError

	Specifies the action a DB2 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 DB2 access service to reject the row containing the error and issue a warning message.
	null specifies the DB2 access service to insert a NULL into the column and issues a warning message (unless the source column is defined as "not null").
	default specifies that the DB2 access service inserts the default date and time values as configured in the DefaultDate and DefaultTime properties into the column (see "DefaultDate" on page 40 and "DefaultTime" on page 41) and issues a warning message.
Comments	The SendWarningMessages property controls whether the DB2 access service returns warning messages to the client application. (See "SendWarningMessages" on page 35.)

DefaultDate

Specifies the value a DB2 access service inserts into columns with date conversion errors when DateTimeConvertError is set to default. (See “DateTimeConvertError” on page 39.)

Note The DefaultDate property value must be within a valid range for the associated data conversion property values.

Syntax	DefaultDate=*yyyy-mm-dd*
Default	1900-01-01
Values	*yyyy-mm-dd* where: • *yyyy* is the year. • *mm* is the month. • *dd* is the day.

DefaultNum

Specifies the value a DB2 access service inserts into columns with numeric conversion errors when NumConvertError is set to default. (See “NumConvertError” on page 41.)

Note The DefaultNum property value must be within a valid range for the associated data conversion property values.

Syntax	DefaultNum=*integer*
Default	0
Values	*integer* is a valid number that replaces the value that caused the conversion error.

DefaultTime

Specifies the value a DB2 access service inserts into columns with time conversion errors when DateTimeConvertError is set to default. (See “DateTimeConvertError” on page 39.)

Note The DefaultTime property value must be within a valid range for the associated data conversion property values.

Syntax	DefaultTime=*hh.mm.ss*
Default	00.00.00
Values	*hh.mm.ss* is the default time, 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 a DB2 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 DB2 access service to reject the row containing the error and issue a warning message.

null specifies the DB2 access service to insert a NULL into the column and issue a warning message.

default specifies the DB2 access service to:

- Insert the default numeric value as configured in the DefaultNum property into the column. (See “DefaultNum” on page 40.)
- Issue a warning message to the client application.

Comments

The SendWarningMessages property controls whether the DB2 access service returns warning messages to the client application. (See “SendWarningMessages” on page 35.)

Datatype Conversion properties

These properties control how a DB2 access service converts target database datatypes to Open Client and Open Server datatypes before sending the data to the client application.

The subsection heading for these properties must appear in the service library configuration file as:

(Datatype Conversion)

BinaryResults
DateResults
DateTimeResults
DecimalResults
FloatResults
GraphicResults
Int2Results
Int4Results
RealResults
TimeResults
TinyIntResults
XNLChar
XNLVarChar

To provide portability across DBMSs, 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 Chapter 4, "Converting Datatypes."

BinaryResults

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 result columns that are described as either CHAR FOR BIT DATA or VARCHAR FOR BIT DATA.

Syntax

```
BinaryResults=[binary | char]
```

Default

binary

Values

binary indicates that results of XNLChar bytes or less are returned as CS_BINARY; those with more than XNLChar bytes or more are returned as CS_IMAGE.

char indicates that results of XNLChar bytes or less are returned as CS_CHAR; those with more than XNLChar bytes or more are returned as CS_TEXT.

DateResults

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 DATE results.

Syntax

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

Default

datetime

Values

With Open Client version 10.0.4 and later, the default display format of CS_DATETIME for character conversions changes from a *long* version to a *short* version:

datetime returns CS_DATETIME. The range of legal values is 1/1/1753 to 12/31/9999, and a precision of 1/300 of a second.:

datetime4 returns CS_DATETIME4, a 4-byte datetime datatype with a range of legal values from January 1, 1900, to June 6, 2079, and a precision of 1 minute.

char_iso returns CS_CHAR, character data in the format yyyy-mm-dd.

char_usa returns CS_CHAR, character data in the format mm/dd/yyyy.

char_eur returns CS_CHAR, character data in the format dd.mm.yyyy.

char_jis returns CS_CHAR, character data in the format yyyy-mm-dd.

char_odbc returns character data in the format yyyy-mm-dd.

DateTimeResults

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 TIMESTAMP results.

Syntax

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

Default

datetime

Values

datetime returns CS_DATETIME, 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 CS_DATETIME4, a 4-byte datetime datatype with a range of legal values from January 1, 1900, to June 6, 2079, and a precision of 1 minute.

char_iso returns CS_CHAR, character data in the format *yyyy-mm-dd-hh.mm.ss.nnnnnn.*

char_usa returns CS_CHAR, character data in the format *mm/dd/yyyy hh:mm* [AM | PM].

char_eur returns CS_CHAR, character data in the format *dd.mm.yyyy hh.mm.ss.*

char_jis returns CS_CHAR, character data in the format *yyyy-mm-dd hh:mm:ss.*

char_odbc returns CS_CHAR, character data in the format *yyyy-mm-dd hh:mm:ss.nnnnnn.*

Comments

You can convert DB2 TIMESTAMP to one of the character formats to retain more precision. CS_DATETIME has less precision (1/300ths of a second) than TIMESTAMP (millionths of a second).

DecimalResults

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 DECIMAL results.

Syntax

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

Default

autoconvert

Values

autoconvert means the DB2 access service chooses the appropriate datatype to return according to the following conversion scheme:

- The DB2 access service returns CS_DECIMAL datatype to Client-Library applications.
- The DB2 access service returns the following values for Client-Library and DB-Library client applications prior to version 10.x:

- If the decimal value has 0 digits to the right of the decimal delimiter and the total number of digits is less than or equal to 9 (scale equals 0 and precision is equal to or less than 9), the DB2 access service returns CS_INT.
- If the decimal value has 0 to 2 digits to the right of the decimal delimiter and the total number of digits is less than or equal to 14 (scale is equal to or less than 2 and precision minus scale is equal to or less than 14), the DB2 access service returns CS_MONEY.
- For any other decimal value, the DB2 access service returns CS_FLOAT.

int returns CS_INT, a 4-byte integer type.

float returns CS_FLOAT, an 8-byte float type.

real returns CS_REAL, a 4-byte float type.

char returns CS_CHAR, a character type.

money returns CS_MONEY, an 8-byte money type.

money4 returns CS_MONEY4, a 4-byte money type.

bcd is valid only if you have columns described in BCD format.

The DB2 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 binary coded decimal (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.

Comments

If the ClientDecimalSeparator property value is a character other than a period (.), set the DecimalResults property value to char to enable the DB2 access service to return decimal results with the correct client decimal delimiter.

FloatResults

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 FLOAT results.

Syntax	FloatResults=[float | real | char]
Default	float
Values	float returns CS_FLOAT, an 8-byte float type.
	real returns CS_REAL, a 4-byte float type.
	char returns CS_CHAR, a character type.

GraphicResults

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 GRAPHIC, VARGRAPHIC, and LONGVARGRAPHIC results.

Syntax	GraphicResults=[*binary* | *char*]
Default	binary
Values	*binary* indicates that results of XNLChar and XNLVarChar configuration property values or less are returned as CS_BINARY; those with XNLChar and XNLVarChar configuration property values or more are returned as CS_IMAGE.
	char indicates that results of XNLChar and XNLVarChar configuration property values or less are returned as CS_CHAR; those with XNLChar and XNLVarChar configuration property values or greater are returned as CS_TEXT.

Int2Results

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 SMALLINT results.

Syntax	Int2Results=[smallint | char]
Default	smallint
Values	smallint returns CS_SMALLINT, a 2-byte integer type.
	char returns CS_CHAR, a character type.

Int4Results

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 INTEGER results.

Syntax	Int4Results=[int | char]
Default	int
Values	int returns CS_INT, a 4-byte integer type.
	char returns CS_CHAR, a character type.

RealResults

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 REAL results.

Syntax	RealResults=[float | real | char]
Default	float
Values	float returns CS_FLOAT, an 8-byte float type.
	real returns CS_REAL, a 4-byte float type.
	char returns CS_CHAR, a character type.

TimeResults

Specifies the Open Client and Open Server datatype to which a DB2 access service converts DB2 TIME results.

Syntax	TimeResults=[datetime | datetime4 | char_iso | char_usa | char_eur | char_jis | char_odbc]
Default	datetime
Values	datetime returns CS_DATETIME, 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 CS_DATETIME4, 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 CS_CHAR, character data in the format *hh.mm.ss*.

char_usa returns CS_CHAR, character data in the format *hh:mm* [AM | PM].

char_eur returns CS_CHAR, character data in the format *hh.mm.ss*.

char_jis returns CS_CHAR, character data in the format *hh:mm:ss*.

char_odbc returns CS_CHAR, character data in the format *hh:mm:ss*.

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, binary, and graphic 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 - 2,147,483,647 (two gigabytes).

Comments

Sybase recommends that this value match the maximum size of the char, binary, and graphic 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 - 2,147,483,647.
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

General server log file information

Each log entry consists of a number of columns of data, each separated by a tab character. These columns appear in the following order:

1. Record type
2. Date/Time
3. Service Library Name, Service Name, or Server Name
4. SPID (Server Process ID), the identifier for the current client connection
5. User ID
6. Application Name
7. Specific Information

   The server defines the first six columns. However, the Specific Information column contains information specified by the logging properties. Logging properties exist at the server, service library, and service levels.

   For detailed information about server properties and the server log file, see the DirectConnect *Server Administration Guide*.

General log statistics information

All statistics properties record some identical types of data. You use these properties independently or in combination to record statistics at whatever level of granularity necessary to perform your analysis (see “Logging properties” on page 49).

The following table describes the log statistics properties.

Table 2-1: Log statistics properties

Property name	Description
LogRequestStatistics	A DB2 access service property that records statistics about individual SQL requests.
LogTransferStatistics	A DB access service property that records statistics about individual transfer requests.
LogConnectionStatistics	AnDB2 access service property that records accumulated statistics about requests made by each client connection. Statistics are recorded when the client disconnects.
LogServiceStatistics	A DB2 access service property that records accumulated statistics about requests made by all connections to this DB2 access service.
LogSvclibStatistics	A DB2 Access Service Library property that records accumulated statistics about requests made by connections to all DB2 access services within this DB2 Access Service Library.

Statistics are in standard format, tab-delimited columns in the server log file.

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

Table 2-2: LogSvclibStatistics 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 DB2 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 DB2 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 DB2 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
Number of successful connections	The total number of successful client connections that occurred during the time period
Maximum number of client connections	The greatest number of client connections at any given time during the time period

Logging properties control whether DB2 access service data is recorded in the server log file. For detailed information about the server log file, see the DirectConnect *Server Administration Guide*.

The subsection heading and the properties must appear in the DB2 Access Service Library configuration file as:

{Logging}

LogConnectionStatistics
LogReceivedSQL
LogRequestStatistics
LogServiceStatistics
LogSvclibStatistics
LogTargetActivity
LogTransferStatistics
LogTransformedSQL

LogConnectionStatistics

Specifies whether the DB2 access service records accumulated statistics about all requests performed by a client 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 DB2 access service. • You use this property to monitor the activity of particular clients.

LogReceivedSQL

Specifies whether the DB2 access service records SQL statements when the statements are received from client applications.

Syntax	LogReceivedSQL=[no | yes]
Default	no
Values	no means the DB2 access service does not record SQL statements when the statements are received. yes means the DB2 access service records SQL statements when the statements are received.

LogRequestStatistics

Specifies whether the DB2 access service records statistics about each SQL request.

Syntax	LogRequestStatistics=[no | yes]
Default	no
Values	no means the DB2 access service does not record statistics about each request. yes means the DB2 access service records statistics about each request.

Comments

- SQL requests are recorded in the server log file when the requests occur.
- You use this property to:
    - Aid performance tuning on a specific type of request
    - Analyze data throughput
    - Monitor the types of requests by users

LogServiceStatistics

Specifies how often the DB2 access service records accumulated statistics about requests made by all connections to the DB2 access service during the reporting interval.

Syntax	LogServiceStatistics=*integer*
Range	0–2147483646
Default	0
Values	*integer* is a number of seconds that specifies how often statistics are recorded in the server log file. A value of 0 specifies that the DB2 access service does not record DB2 access service statistics.

Comments

- You use this property to:
    - Monitor the load on a particular DB2 access service
    - Monitor usage of a particular DB2 access service

      For example, use this property to determine the time of day that the DB2 access service has the greatest usage.

- If the property value is greater than 0, the DB2 access service records the DB2 access service statistics shown in Table 2-2 on page 51.

LogSvclibStatistics

Specifies how often the DB2 Access Service Library records accumulated statistics about requests made by connections to all DB2 access services associated with this DB2 Access Service Library during the reporting interval.

Syntax	LogSvclibStatistics=*integer*
Range	0–2147483646
Default	0
Values	*integer* is a number of seconds. A value of 0 specifies that the DB2 Access Service Library does not record statistics in the server log file.

Comments

- 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. For example, if you configure LogSvclibStatistics for 60 seconds, you then configure LogServiceStatistics for either 60, 120, 180 seconds and so on. If you use DirectConnect Manager to change these two property values, set the LogSvclibStatistics property last for better synchronization.
- You use this property to:
  - Monitor load on the entire DB2 Access Service Library
  - Monitor load on the target database through this DirectConnect server
- If the LogSvclibStatistics property value is greater than 0, the DB2 Access Service Library records totals of the statistics for all DB2 access services in the DB2 Access Service Library.

  For example, if the DB2 Access Service Library contains DB2 access services named Service A and Service B, the data recorded for the maximum number of client connections would contain the total of the Service A maximum number of client connections plus the Service B maximum number of clients connections at any given time during the time period.

  The following table shows DB2 Access Service Library statistics.

LogTargetActivity

Specifies whether the DB2 access service records DB2 access service interactions with the target database.

Syntax

LogTargetActivity=[no | yes]

Default

no

Values

no means the DB2 access service does not record DB2 access service interactions with the target database.

yes means the DB2 access service records the following DB2 access service interactions with the target database:

- Login
- Logout
- Requests sent
- Results received

LogTransferStatistics

Specifies whether the DB2 access service records statistics about transfers.

Syntax

LogTransferStatistics=[no | yes]

Default

no

Values

no means the DB2 access service does not record transfer statistics.

yes means the DB2 access service records statistics about each transfer.

Comments

The DB2 access service records transfer statistics as 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 DB2 access service.
Transfer setup time	The elapsed time in seconds from when the DB2 access service receives a transfer statement until it issues the source select against the secondary database. This includes connection time to the secondary database.
Source DBMS processing time	The elapsed time in seconds from when the DB2 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. This includes the time the DB2 access service takes to obtain datatype information for target columns.
Time to transfer rows	The elapsed time in seconds from when the source database returns the first row to the DB2 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 DB2 access service receives the transfer statement until the last row of the transfer 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.
Command type	This is either bulk transfer or template transfer.

LogTransformedSQL

Specifies whether the DB2 access service records SQL as it is transformed and sent to the target database.

Syntax

LogTransformedSQL=[no | yes]

Default

no

Values

no means the DB2 access service does not record SQL as it is transformed and sent to the target database.

yes means the DB2 access service records SQL as it is transformed and sent to the target database.

Target Interaction properties

These properties control how a DB2 access service interacts with the target database.

The subsection heading and the properties must appear in the service library configuration file as:

{Target Interaction}

Allocate
CloseOnEndTran
PasswordRequired
QuotedStringDelimiter
SQLTransformation
StopCondition
TargetDebug
TargetDecimalSeparator
TargetHasMixedData

Allocate

Controls when a DB2 access service allocates and deallocates conversations with the target database system.

Syntax	Allocate=[connect | request]
Default	connect
Values	connect indicates that a DB2 access service allocates the conversation when the client connects and holds it open for the duration of the client connection. request indicates that a DB2 access service allocates a new conversation each time the client application sends a request and deallocates the conversation after each request.
Comments	See Chapter 5, “Understanding the Request Process,” for more information on request processing.

APPCSecurity

Controls how a DB2 access service handles advanced program-to-program communications (APPC) security when it allocates an LU 6.2 conversation.

Syntax	APPCSecurity=[pgm | none | same]
Default	pgm
Values	pgm specifies that a DB2 access service sends both a user ID and password to the target database system when it establishes communications.
	none specifies that a DB2 access service sends neither a user ID nor a password to the target database system when it establishes communications.
	same specifies that a DB2 access service sends a user ID, but not a password to the target database system when it establishes communications.

CloseOnEndTran

Determines how cursors behave when a commit is executed.

Syntax	CloseOnEndTran= [on | off]
Default	on
Values	on (the default) causes all cursors to be closed.
	off causes cursors to remain open at their positions when the commit is executed.
Comments	Returns how cursors behave when a commit is executed.

PasswordRequired

Specifies whether a DB2 access service requires a user-supplied password to allocate the conversation with the database. The DB2 access service sends the password to the database.

Syntax	PasswordRequired=[no | yes]
Default	no
Values	no indicates that a DB2 access service attempts to connect to the database whether or not there is a password. However, if the database requires a password for login and none is provided, the database sends an error message to the client application.

	yes indicates that a DB2 access service checks for a password before it makes the APPC connection. If it does not find a password, the DB2 access service does not attempt to make a connection and sends an error message to the client application.
Comments	The PasswordRequired property has no tie with the APPCSecurity property.

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.
Comments	This property should match the setting configured during DB2 installation.

SQLTransformation

	Specifies the mode a DB2 access service uses for SQL transformation.
Syntax	SQLTransformation=[passthrough | sybase]
Default	passthrough
Values	passthrough indicates that a DB2 access service sends 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 indicates that a DB2 access service performs SQL transformation on selected statements. It also allows the use of multi-part table names with the view command in SQL statements.
Comments	• See “SQL transformation modes” on page 107 for detailed information about passthrough and sybase modes. • For backward compatibility only, the DB2 access service also accepts the following parameters: [db2 | tsql0 | tsql1 | tsql2]

Setting to db2 or tsql0 is the same as setting to passthrough. For a description of tsql1 and tsql2, which are no longer supported and are identified here only for backward compatibility, refer to the *MDI Database Gateway User's Guide* for your DB2 database and platform.

StopCondition

Specifies under what conditions a DB2 access service stops processing.

Syntax	StopCondition=[error | none | warning]
Default	error
Values	error indicates that a DB2 access service stops processing results only when an error occurs.
	none indicates that a DB2 access service does not stop processing results when errors or warnings occur.
	warning indicates that a DB2 access service stops processing results when an error or warning occurs.

TargetDebug

Specifies whether the mainframe access module (AMD2) runs a trace of the transaction program on the mainframe.

Warning! Do not change the default setting unless Sybase Technical Support instructs you to do so.

Syntax	TargetDebug=[none | statistics | time | trace]
Default	none
Values	none indicates that AMD2 does not run a trace except for an abend.
	statistics indicates that AMD2 traces statistics.
	time indicates that AMD2 traces absolute times at specific processing points.
	trace indicates that AMD2 runs a full trace including statistics and time information.

TargetDecimalSeparator

Specifies the character the target database uses to separate decimal numbers for presentation purposes. The target database does not store the target decimal delimiter character.

Syntax	TargetDecimalSeparator=[. | ,]
Default	. (period)
Values	A period (.) indicates that the target database uses a period as the decimal delimiter. A comma (,) indicates the target database uses a comma as the decimal delimiter.

TargetHasMixedData

Indicates whether the DB2 access service allows character strings returned in the sp_columns catalog stored procedure (CSP) result set to contain a mixture of SBCS and DBCS characters. For information about sp_columns, see sp_columns on page 183.

Syntax	TargetHasMixedData=[no | yes]
Default	no
Values	no means that character strings returned by sp_columns cannot contain a mixture of SBCS and DBCS characters. yes means that character strings returned by sp_columns can contain a mixture of SBCS and DBCS characters.
Comments	The TargetHasMixedData property value must match the MIXED DATA option on the target database.

Tracing properties

Tracing properties control whether DB2 access service data is written to the server trace file. For detailed information about the server trace file, see the DirectConnect *Server Administration Guide*.

The subsection heading and the properties must appear in the service library configuration file as:

{Tracing}

TraceEvents
TraceHostCom
TraceInterface
TraceTarget

Because tracing degrades performance, use tracing only when Sybase Technical Support instructs you to configure specific tracing properties.

Warning! To provide Sybase Technical Support with all necessary 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 file name and with an "_old" extension (<*filename*>*_old*). See the DirectConnect *Server Administration Guide* for suggestions for deleting or backing up old log and trace files.

TraceEvents

Specifies whether the DB2 access service traces the event handler layer of the DB2 Access Service Library.

Syntax	TraceEvents=[no | yes]
Default	no
Values	no means the DB2 access service does not trace the event handler layer.
	yes means the DB2 access service traces the event handler layer.
Comments	The DB2 access service traces Open Server calls such as connect, disconnect, language, and cursor events.

TraceHostCom

Specifies whether the DB2 access service traces the host communication layer of the DB2 Access Service Library.

Syntax	TraceHostCom=[no | yes]
Default	no
Values	no means the DB2 access service does not trace the host communication layer.

yes means the DB2 access service traces the host communication layer.

Comments

The DB2 access service traces all communications to and from the platform supporting the target database (LU 6.2 and TCP/IP communications).

TraceInterface

Specifies whether the DB2 access service traces the interface layer of the DB2 Access Service Library.

Syntax

TraceInterface=[no | yes]

Default

no

Values

no means the DB2 access service does not trace the interface layer.

yes means the DB2 access service traces the interface layer.

Comments

- The DB2 access service traces all activity related to interfacing the DB2 Access Service Library and the target database such as Open Client DB-Library, and Open Client CT-Library, and TDS Library activity.
- TDS Library activity is written to a separate trace file named *tds.trc* which resides on the server in the same directory as the server trace file.
- If you change the TraceInterface property value from no to yes using DirectConnect Manager, the Allocate property setting determines when TDS tracing takes effect for the client connection:
  - If the Allocate property is set to request, the DB2 access service begins tracing to the *tds.trc* file with the next request.
  - If the Allocate property is set to connect, the client application must log out and log back in for the DB2 access service to begin TDS tracing.
- The TDS Library activity trace file does not wrap and needs to be deleted periodically. You delete the *tds.trc* file the same as you would the server trace file. See the DirectConnect *Server Administration Guide*.

TraceTarget

Specifies whether the DB2 access service traces the implementation layer of the DB2 Access Service Library.

Syntax

TraceTarget=[no | yes]

Default	no
Values	no means the DB2 access service does not trace the implementation layer. yes means the DB2 access service traces the implementation layer.
Comments	The DB2 access service traces all database-specific target activity within the DB2 Access Service Library.

Transfer properties

These properties control how a DB2 access service performs transfer processing.

The subsection heading for these properties must appear in the service library configuration file as:

{Transfer}

BulkCommitCount
TransferBatch
TransferErrorAction
TransferErrorCount

BulkCommitCount

Specifies the number of rows sent in a bulk transfer before a commit is issued.

Syntax	BulkCommitCount=*integer*
Range	0–32767
Default	0
Values	*integer* is the number of rows sent in a bulk transfer. If the value is 0, the DB2 access service issues a commit at the end of the transfer. If the value is non-zero, the DB2 access service issues a commit after every *n* rows. For example, if BulkCommitCount is set to 50, a commit is issued after every 50 rows.
Comments	• The BulkCommitCount property is useful when you make large transfers *to* Adaptive Server and, as a result, run out of page locks on SQL Server. Issuing a commit after every *n* rows clears the page locks.

- With this property, after rows are committed, they cannot be rolled back. Therefore, if you set BulkCommitCount to a non-zero value and a transfer fails, the DB2 access service rolls back only the batch containing data since the last commit was issued.
- Setting BulkCommitCount can reduce bulk transfer performance when transferring data *from* the secondary database into DB2. This occurs because the DB2 access service sends as many rows as possible to DB2 in a single transfer but never more than the BulkCommitCount setting. If setting BulkCommitCount reduces transfer performance to DB2, try increasing the BulkCommitCount setting until performance is similar to that achieved with BulkCommitCount=0.
- For more information on bulk transfer copy, see Chapter 9, “Using Bulk Copy Transfer.”.

TransferBatch

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

Syntax	TransferBatch=*integer*
Range	0–32767
Default	1
Values	*integer* is a number of destination-template clauses. If the value is 0, the DB2 access service sends all statements that fit in the request buffer.
Comments	• The DB2 access service accumulates the designated number of destination templates in its request buffer before executing each request. • 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 bulk copy transfer or destination-template transfer before processing stops.

Syntax	TransferErrorCount=*integer* where *integer* is a number of error rows.
Range	0–32767
Default	0
Values	*integer* is a number of rows with errors. A value of 0 means transfers do not stop processing regardless of the number of errors encountered.
Comments	For detailed information about how the TransferErrorCount property affects transfers, see “TransferErrorCount property” on page 146.

CHAPTER 3

Querying and Setting Operating Values

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

This chapter includes 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:

- A configuration property value
- Information about the processing state of the current connection
- The current value for a configuration property resulting from a set statement

SQL transformation mode

A client application can query all global variables regardless of the SQL transformation mode in effect. However, the global variable statement must be the only statement in a request. For more information about SQL transformation modes, see “SQL transformation modes” on page 107.

Syntax

Global variables are preceded by two “at” (@@) symbols and are not case sensitive.

To query a global variable, issue a SQL statement in the following form:

select @@*variable_name*

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

The access service returns the configuration property value or the processing information for the current connection.

For example, the SQL statement select @@Allocate returns the Allocate configuration property value of either connect or request.

Issuing set statements

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

SQL transformation mode

A client can set values regardless of the SQL transformation mode in effect. However, the set statement must be the only statement in a request. For more information about SQL transformation modes, see “SQL transformation modes” on page 107.

Syntax

DB2 access service set statements are not case sensitive.

To set an DB2 access service configuration property value or processing value for the current connection, issue a set statement in the form:

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.

For example, the statement set Allocate request sets the Allocate configuration property value to request.

Querying and setting properties

Some configuration property values can be queried or set using global variables and set statements. These properties appear grouped by category in the following sections:

- ACS (DB2 access service) required properties
- Target Interaction properties
- Client Interaction properties
- Catalog Stored Procedure properties
- Datatype Conversion properties
- Data Conversion Error properties
- Transfer properties

Configuration properties, global variables, and set statements are not case sensitive.

For explanations of configuration properties, see Chapter 2, “Creating and Configuring DB2 Access Services.”

ACS (DB2 access service) required properties

The following table shows DB2 access service required properties and associated global variables and set statements.

Table 3-1: DB2 Access Service required properties

Configuration property	Global variable and set statement
DefaultClientCodeset	select @@DefaultClientCodeset
	set DefaultClientCodeset *codeset*
DefaultTargetCodeset	select @@DefaultTargetCodeset
	No set statement

Target Interaction properties

The following table shows Target Interaction properties and associated global variables and set statements.

Table 3-2: Target Interaction properties

Configuration property	Global variable and set statement
Allocate	select @@Allocate
	set Allocate {connect | request}
SQLTransformation	select @@SQLTransformation
	set SQLTransformation {passthrough | sybase}
StopCondition	select @@StopCondition
	set StopCondition {error | none | warning}
TargetDebug	select @@TargetDebug
	set TargetDebug {none | statistics | time | trace}
TargetDecimalSeparator	select @@TargetDecimalSeparator
	No set statement

Client Interaction properties

The following table shows client interaction properties and associated global variables and set statements.

Table 3-3: Client Interaction properties

Configuration property	Global variable and set statement
ClientDecimalSeparator	select @@ClientDecimalSeparator set ClientDecimalSeparator *char*
GatewayCompatible	select @@GatewayCompatible No set statement
MaxResultSize	select @@MaxResultSize set MaxResultSize integer
MaxRowsReturned	select @@MaxRowsReturned set MaxRowsReturned integer
MaxSvcConnections	select @@MaxSvcConnections No set statement
Quoted_Identifier	select @@quoted_identifier set Quoted_Identifier {no | yes}
SendWarningMessages	select @@SendWarningMessages set SendWarningMessages {no | yes}
ServiceDescription	select @@ServiceDescription No set statement
SvclibDescription	select @@SvclibDescription (This global variable applies to the DB2 Access Service Library as a whole.) No set statement
TextSize	select @@TextSize set TextSize integer
TransactionMode	select @@TransactionMode set TransactionMode {short | long}
Version	select @@Version No set statement Returns the current version string in effect. To find the default version, see sp_helpserver on page 211.

Catalog Stored Procedure properties

The following table shows Catalog Stored Procedure (CSP) properties and associated global variables and set statements.

Table 3-4: Catalog Stored Procedure properties

Configuration property	Global variable and set statement
CSPCatalogQualifier	select @@CSPCatalogQualifier set CSPCatalogQualifier
CSPDBName	select @@CSPDBName set CSPDBName {NULL |dbname}
CSPExclusions	select @@CSPExclusions set CSPExclusions {none | user | nonauth | nonauthpublic}
CSPIncludeAlias	select @@CSPIncludeAlias set CSPIncludeAlias {no | yes}
CSPIncludeSynonym	select @@CSPIncludeSynonym set CSPIncludeSynonym {no | yes}
CSPIncludeSystem	select @@CSPIncludeSystem set CSPIncludeSystem {no | yes}
CSPIncludeTable	select @@CSPIncludeTable set CSPIncludeTable {yes | no}
CSPIncludeView	select @@CSPIncludeView set CSPIncludeView {yes | no}
CSPQualByDBName	select @@CSPQualByDBName set CSPQualByDBName {no | yes}
DatatypeInfo	select @@DatatypeInfo set DatatypeInfo {transact | target}

Datatype Conversion properties

The following table shows Datatype Conversion properties and associated global variables and set statements.

Table 3-5: Datatype Conversion properties

Configuration property	Global variable and set statement
BinaryResults	select @@BinaryResults set BinaryResults {binary | char}
DateResults	select @@DateResults set DateResults {datetime | datetime4 | char_eur | char_iso | char_jis | char_usa}
DateTimeResults	select @@DateTimeResults set DateTimeResults {datetime | datetime4 | char_eur | char_iso | char_jis | char_usa}
DecimalResults	select @@DecimalResults set DecimalResults {autoconvert | char | int | real | float | money | money4 | bcd}
FloatResults	select @@FloatResults set FloatResults {float | real | char}
GraphicResults	select @@GraphicResults set GraphicResults {binary | char}
Int2Results	select @@Int2Results set Int2Results {smallint | char}
Int4Results	select @@Int4Results set Int4Results {int | char}
RealResults	select @@RealResults set RealResults {float | real | char}
TimeResults	select @@TimeResults set TimeResults {datetime | datetime4 | char_eur | char_iso | char_jis | char_usa}

Data Conversion Error properties

The following table shows Data Conversion Error properties and associated global variables and set statements.

Table 3-6: Data Conversion Error properties

Configuration property	Global variable and set statement
CharConvertError	select @@CharConvertError set CharConvertError {reject | truncate}
DateTimeConvertError	select @@DateTimeConvertError set DateTimeConvertError {reject | null | default}
NumConvertError	select @@NumConvertError set NumConvertError {reject | null | default}

Transfer properties

The following table shows Transfer properties and associated global variables and set statements.

Table 3-7: Transfer properties

Configuration property	Global variable and set statement
BulkCommitCount	select @@BulkCommitCount set BulkCommitCount *integer*
TransferBatch	select @@TransferBatch set TransferBatch *integer*
TransferErrorCount	select @@TransferErrorCount set TransferErrorCount *integer*

Querying and setting processing values

The following table shows global variables and set statements used to query and set processing values for the client connection.

Table 3-8: Global variables and set statements for processing values

Global variable and set statement	Description
select @@AllResults	Returns group datatype conversions.
set AllResults {autoconvert | char}	Sets group datatype conversions: • autoconvert sets all datatype conversions to the default values. • char sets all datatype conversions to CS_CHAR.
select @@CloseOnEndTran	Returns how cursors behave when a commit is executed.
set CloseOnEndTran {on | off}	Determines how cursors behave when a commit is executed: • on (the default) causes all cursors to be closed. • off causes cursors to remain open at their positions when the commit is executed.
select @@Connections No set statement	Returns the current number of connections to this DB2 access service.
select @@DefaultedRowCount No set statement	Returns the number of rows the DB2 access service returned with default values substituted for data conversion errors.
select @@Error No set statement	Reflects the message number of each event. A successful event returns a 0 (zero).
select @@MainframeVersion No set statement	Returns the version of MainframeConnect for DB2.
select @@noexec set noexec {on | off}	Returns whether metadata are returned for a select statement result set. Determines whether metadata are returned for a select statement result set: • on sets the DB2 access service to handle subsequent select statements such that no results rows are returned, but result set metadata are available. • off (the default) returns the DB2 access service to the normal mode of returning result sets.
select @@RejectedRowCount No set statement	Returns the number of rows rejected by the DB2 access service due to data conversion errors.

Global variable and set statement	**Description**
select @@RowCount No set statement	Returns the number of rows affected by the last SQL statement processed.
select @@ServiceName No set statement	Returns this DB2 access service name.
select @@spid No set statement	Returns a unique positive integer identifier (server process ID) for the current client connection.
select @@TargetError No set statement	Returns the message number of the last target database error.

CHAPTER 4 Converting Datatypes

This chapter describes datatype conversions between DB2 and Open Client and Open Server. It chapter contains the following topics:

Converting target datatypes to Open Client and Open Server datatypes

When you retrieve data from the target database, the DB2 access service converts the target data to default Open Client and Open Server datatypes for delivery to the client application. You can configure datatype conversions by:

- Changing the datatype conversion property values when you configure an access service. (For a complete description of datatype conversion configuration properties and values, see “Datatype Conversion properties” on page 42.)
- Setting the conversions required by a particular client application by issuing set statements from the client application.

Issuing set statements to change datatype conversions

You can issue set statements to change one or more datatype conversions for the current client connection. These settings remain in effect only for the duration of the current client connection or until you issue another set statement.

For more information about set statements, see Chapter 3, "Querying and Setting Operating Values."

Single datatype conversions

You can set the datatype conversion of a specific datatype by issuing a set statement that contains the specific datatype configuration property and value.

For example, to set the datatype conversion of the DB2 FLOAT datatype to Open Client and Open Server CS_CHAR datatype, issue the following statement:

```
set FloatResults char
```

where:

- *FloatResults* is the datatype conversion configuration property that maps to the DB2 FLOAT datatype
- char is the property value that maps to the Open Client and Open Server datatype CS_CHAR.

Group datatype conversions

To change a group of datatype conversion settings to the default setting for each datatype or Open Client and Open Server CS_CHAR, issue the following set statement:

```
set AllResults [autoconvert | char ]
```

Character representations

The following table lists the character representations that a DB2 access service returns to the client application when you choose char as the output type. The table includes configuration property values for date and time character forms. For example, DB2 DATE type can convert to Open Client and Open Server CS_CHAR type in one of the four character formats shown in the table.

Table 4-1: Character representations

DB2 input type	Open Client and Open Server character form returned to the client application
DECIMAL (scale = 0)	±n
DECIMAL (scale > 0)	±n.n
DECIMAL (scale = precision)	±.n
DATE	char_eur format *dd.mm.yyyy* char_iso format *yyyy-mm-dd* char_jis format *yyyy-mm-dd* char_usa format *mm/dd/yyyy*
REAL	±n.nnnnnnnE±nn
FLOAT	±n.nnnnnnnnnnnnnnnE±nn
INT	±nnnnnnnnnn
SMALLINT	±nnnnn
TIME	char_eur format *hh.mm.ss* in 24-hour clock time char_iso format *hh.mm.ss* in 24-hour clock time char_jis format *hh:mm:ss* in 24-hour clock time char_usa format *hh:mm* [AM | PM] in 12-hour clock time
TIMESTAMP	char_eur format *dd.mm.yyyy hh.mm.ss* char_iso format *yyyy-mm-dd-hh.mm.ss.nnnnnn* char_jis format *yyyy-mm-dd hh:mm:ss* char_usa format *mm/dd/yyyy hh.mm* [AM | PM]

Data conversion errors

A data conversion error occurs when a value is out of the valid range. For example, if a DB2 source column with a DATE datatype has a value of Jan 1, 1899, and the DB2 access service datatype conversion property DateResults is set to datetime4, a value error occurs because the DB2 DATE value cannot be expressed in an Open Client and Open Server CS_DATETIME4 datatype.

Note When the appropriate DateTimeConvertError or NumConvertError property value is reject and DB2 returns results with data values that are out of range for the associated DB2 access service data conversion property value, the DB2 access service inserts a header row into the result set immediately after the rejected row. This results in an additional header row for every rejected row in the result set.

Configuration properties control the behavior of an access service when it encounters data conversion errors.

For example, when converting DB2 DATE or TIME values to Open Client and Open Server CS_DATETIME values, an access service can insert default information, which is configured by data conversion error properties, for the unknown date or time.

For a description of data conversion error properties and associated values, refer to "Data Conversion Error properties" on page 38.

The following table indicates conditions that cause value errors.

Table 4-2: Conversions that cause errors

DB2 datatype & Open Client and Open Server conversion	DB2 value
FLOAT, REAL, or DECIMAL and FloatResults value is real	Greater than 3.402823466E38 or less than -3.402823466E38
DECIMAL, FLOAT, or REAL and DecimalResults value is money	Greater than 922337203685477.5807 or less than -922337203685477.5808
DECIMAL, FLOAT, REAL or INT and DecimalResults value is money4	Greater than 214748.3646 or less than -214748.3647
DATE DateResults value is datetime	DATE value is less than January 1, 1753
DATE DateResults value is datetime4	DATE value is less than January 1, 1900, or greater than June 6, 2079
TIMESTAMP and DateTimeResults value is datetime	TIMESTAMP year is less than 1753
TIMESTAMP and DateTimeResults value is datetime4	TIMESTAMP date is less than January 1, 1900, or greater than June 6, 2079

FLOAT and REAL conversion

DB2 FLOAT and REAL datatypes have approximately the same precision but different ranges from the corresponding Open Client and Open Server types. Values that are valid in Open Client and Open Server may be out of range for DB2. The following table shows the approximate ranges for DB2 FLOAT and REAL datatypes, and Open Client and Open Server CS_FLOAT and CS_REAL datatypes.

Table 4-3: Ranges for DB2 and Open Client and Open Server FLOAT and REAL

DB2 datatype	Approximate range	Open Client and Open Server datatype	Approximate range
FLOAT	-7.2E75 to +7.2E75	CS_FLOAT	-1.7977E308 to +1.7977E308
REAL	-7.2E75 to +7.2E75	CS_REAL	-3.4028E38 to +3.4028E38

Real number testing

In DB2, real numbers (datatypes REAL, FLOAT, DOUBLE, and DOUBLE PRECISION) are stored as approximations. This means that equality tests on real numbers may not locate all rows expected. The recommended way of testing for a real number value is to use a range test. For example, if you want to locate all rows where the column “A” has a value of 1.2345678, issue the following select statement:

```
select.....from.....where A between 1.234567 and 1.234568
```

Converting Open Client and Open Server datatypes to target datatypes

A DB2 access service converts or performs SQL transformation on incoming data it receives in a client request when the incoming data include:

- Data values embedded as strings within the text of select, insert, delete, update, and execute language commands
- Data values as parameters of RPC, cursor, or dynamic SQL commands
- Datatype names as part of create table or alter table commands

Data values embedded as strings

This section describes datatype conversion and SQL transformation of data values embedded as strings in language commands.

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.

Datatype conversion

When a DB2 access service receives data values embedded in strings, it does not automatically convert the incoming data values. The client application must format the strings correctly with valid datatypes before sending them to the target database. The DB2 access service provides a string template for the target datatypes. To find the correct datatype, the client application receives this string template for the target datatypes from the sp_columns CSP.

SQL transformation

When the access service receives a SQL command that has embedded data values, the SQL transformation mode in effect determines whether or not any transformation is applied to these values:

- If the access service is in passthrough mode, it does not perform transformation.
- If the DB2 access service is in sybase mode, it performs the following transformation:
  - Removes the currency symbol from money constants
  - Transforms quoted strings to quoting conventions specific to the target DBMS

For more information about SQL transformation modes, see “SQL transformation modes” on page 107.

Note Datatype constants are transformed only as shown in the preceding paragraphs. When passing datatype constants, the client must ensure that the constants are in the proper format required by the target DBMS.

Data values received from the client as parameters

This section describes how the DB2 access service handles data values received from the client as parameters of RPC, cursor, or dynamic SQL commands.

Note Only CT-Library clients can issue cursor or dynamic commands. For DB-Library clients, this section applies only to RPC commands.

Default datatype conversion

When a client application sends a parameter description as part of an RPC command, cursor command, or dynamic SQL command, the access service automatically 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.

Client-specified datatype conversion

Some Open Client and Open Server datatypes do not directly map to target datatypes. For example, Open Client and Open Server CS_DATETIME and CS_DATETIME4 datatypes do not directly map to DB2 DATE, TIME, or TIMESTAMP datatypes.

When defaults are not appropriate for these datatype conversions, the CT-Library client can specify the intended DB2 datatype by using the usertype field in the parameter’s CS_DATAFMT structure. The client fills in the CS_DATAFMT structure and a pointer to the structure is passed to the ct_param Open Client function call.

The DB2 access service then converts the Open Client and Open Server datatype to the precise target datatype.

DB-Library clients cannot take advantage of this feature and are limited to the default datatype conversions.

usertype values

The usertype field of the CS_DATAFMT structure is a 32-bit integer.

To find the usertype value, the client application executes sp_columns to obtain a description of the REMOTE_DATA_TYPE column from the sp_columns result set. The REMOTE_DATA_TYPE column returns the integer ID of the ODBC (target) datatype.

The usertype value returned by sp_columns uses various fields in the 32-bit integer for flags, precision and scale, or length. The least significant byte of the value specifies to which target datatype to convert the parameter. The client application places the value in the usertype field. If a 0 (zero) value is placed in the usertype field, the default conversion applies.

For more information

For more information about the CS_DATAFMT structure, ct_param Open Client function call, and usertype field, refer to *Open Client Client-Library/C Reference Manual*. For more information about sp_columns, see sp_columns on page 183.

Parameters related to graphic columns

When you configure an access service for an SJIS or EUCJIS character set, the access service performs double-byte character set (DBCS) translation on character parameters, including those related to graphic columns. The access service translates the contents of graphic columns to Kanji characters.

When you configure an access service for a single-byte character set (SBCS) code page, parameters related to graphic columns must be described as binary types. The access service stores binary data in the column without translation and returns the data exactly as it was sent. The number of bytes of data supplied for a binary parameter must be even.

Datatype conversions from Open Client and Open Server to DB2

The following table lists the possible conversions from Open Client and Open Server datatypes to DB2 datatypes for parameters.

Table 4-4: Datatype conversions from Open Client and Open Server to DB2

Open Client and Open Server datatype	DB2 datatype
CS_BINARY	CHAR FOR BIT DATA (default), VARCHAR FOR BIT DATA, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC
CS_LONGBINARY	CHAR FOR BIT DATA (default), VARCHAR FOR BIT DATA, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC
CS_VARBINARY	CHAR FOR BIT DATA (default), VARCHAR FOR BIT DATA, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC
CS_BIT	SMALLINT
CS_CHAR	CHAR
CS_VARCHAR	VARCHAR
CS_LONGCHAR	CHAR (default), VARCHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC
CS_DATETIME	TIMESTAMP (default), DATE, TIME
CS_DATETIME4	TIMESTAMP (default), DATE, TIME
CS_TINYINT	SMALLINT (all numeric types)
CS_SMALLINT	SMALLINT (all numeric types)
CS_INT	INT (all numeric types)
CS_DECIMAL	DECIMAL (all numeric types)
CS_NUMERIC	DECIMAL (all numeric types)
CS_FLOAT	FLOAT (all numeric types)
CS_REAL	REAL (all numeric types)
CS_MONEY	DECIMAL (all numeric types)
CS_MONEY4	DECIMAL (all numeric types)
CS_TEXT	LONG VARCHAR
CS_IMAGE	LONG VARGRAPHIC

The default DB2 type correspondence does not have to be exact. For example, DB2 accepts either CS_CHAR or CS_VARCHAR for either a DB2 CHAR or VARCHAR column. It also accepts any numeric type for any numeric column.

Datatype names

An access service receives datatype names as part of create table or alter table commands. If the DB2 access service is in passthrough mode, the datatype names are not modified. If the DB2 access service is in sybase mode, Open Client and Open Server datatype names are converted to the target-specific datatype names that correspond to the Open Client and Open Server datatypes. A target database might not be able to support all Open Client and Open Server datatypes, but permits conversion to an equivalent or compatible datatype. For example, the Open Client and Open Server CS_MONEY datatype can be converted to a numeric (19,4) or equivalent datatype.

For more information about passthrough mode and sybase mode, see "SQL transformation modes" on page 107.

Corresponding Open Client and Open Server datatypes and Adaptive Server datatypes

The following table shows corresponding Open Client and Open Server and Adaptive Server datatypes.

Table 4-5: Open Client and Open Server and Adaptive Server datatypes

Open Client and Open Server datatype	Corresponding Adaptive Server datatype
CS_BINARY	binary, varbinary
CS_LONGBINARY	NONE
CS_BIT	bit
CS_CHAR	char
CS_VARCHAR	varchar
CS_LONGCHAR	NONE
CS_DATETIME	datetime
CS_DATETIME4	smalldatetime
CS_TINYINT	tinyint
CS_SMALLINT	smallint
CS_INT	int
CS_DECIMAL	decimal
CS_NUMERIC	numeric
CS_FLOAT	float
CS_REAL	real

Open Client and Open Server datatype	Corresponding Adaptive Server datatype
CS_MONEY	money
CS_MONEY4	smallmoney
CS_TEXT	text
CS_IMAGE	image

CHAPTER 5

Understanding the Request Process

This chapter presents an overview of the request process, including Application Program Interface (API) procedure calls and how they interact with the DB2 access service on a command-based level.

This chapter contains the following topics:

Request types

The DB2 access service processes the following types of requests:

- SQL statements:
  - As a language command
  - As a cursor command (CT-Library only)
  - As a dynamic command
- Catalog stored procedure (CSP) requests
- Remote procedure call (RPC) requests
- Configuration property statements, such as:
  - set statement with a configuration property
  - select configuration property value *(@@global variable)*
- transfer statements

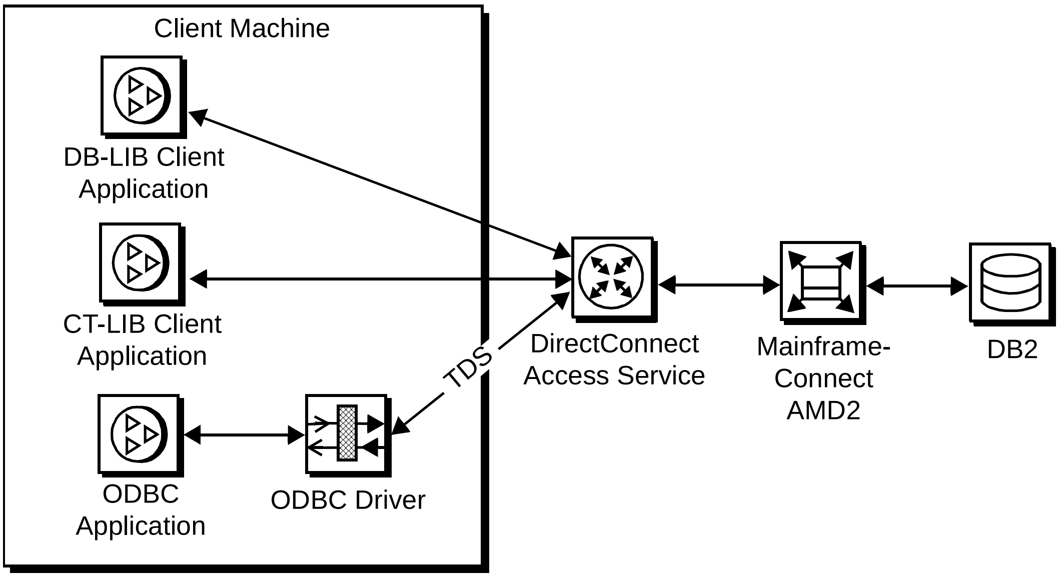
Request processing flow

Request processing follows these steps:

1. The client application issues a request (for example, a select statement).
2. The client API (for example, CT-Library or ODBC) receives the request and sends it to the DB2 access service.
3. The DB2 access service receives the request, transforms it (if needed), and executes the request through MainframeConnect for DB2.
4. After the request processes, the DB2 access service converts target datatypes to Open Server datatypes and returns the results to the client application.
5. The client application disconnects from the DB2 access service.

The following figure shows the processing flow of the APIs through DirectConnect to DB2.

Figure 5-1: Processing flow through API to DB2


Client Machine
DB-LIB Client Application
CT-LIB Client Application
ODBC Application
ODBC Driver
TDS
DirectConnect Access Service
Mainframe-Connect AMD2
DB2

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.*

API calls

This section identifies the client API procedure calls and provides the client programming references:

- ODBC API where the client is using an ODBC application. Client programming is done using the ODBC API following the standard Sybase Open Client methodology.

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

- CT-Library API where client programming is done using the CT-Library API following the standard Open Client methodology.

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

Note For information about specific APIs, see the Open Client *Client-Library/C Reference Manual*.

Managing transactions

To fully understand how properties in the DB2 access service interact to control the processing flow, you should first understand the following concepts:

- Request
- Unit of work
- Short and long transactions

These concepts are discussed in the following section.

Request

A request is one or more database operations sent by the client application as one unit to the database. (For the DB2 access service, a database operation is usually a SQL statement.) During a request, the client application gives up control to the 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. Depending on the setting of the TransactionMode property, either the client application or the DB2 access service issues the commit statement.

For more information about the StopCondition property, see "StopCondition property" on page 94. For the conditions under which the DB2 access service stops processing, see "StopCondition" on page 60.

Transactions

For the DB2 access service, a transaction is equivalent to a unit of work (one or more requests). A transaction can span many requests.

Depending on DB2 access service property settings, the TransactionMode property that governs transaction behavior can be set to either short or long.

Short transactions

When short transactions are in effect, the DB2 access service is responsible for controlling the commitment of requests. After sending the request to the database, the DB2 access service automatically issues one of the following:

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

Note While in short transaction mode, the request is a unit of work.

A begin transaction phrase can affect the behavior of a short transaction. If it receives a begin transaction phrase, the DB2 access service:

- Triggers a commit of all previous statements in the request.
- Temporarily sets TransactionMode to long, although a select @@TransactionMode command returns an answer of short mode. While in temporary long mode, if the batch processes successfully, the SQL statements are committed.

- Stays in temporary long transaction mode until a unit of work is completed with a commit or rollback. Then, the DB2 access service reverts TransactionMode to short.

Note Do not change configuration properties while the request is in temporary long mode. Once a commit or rollback occurs, the temporary long mode reverts to short mode.

Long transactions

When long transactions are in effect, the client application is responsible for controlling when the transaction ends (by issuing either a commit or a rollback).

If the DB2 access service encounters a begin transaction phrase in a request, the phrase is ignored because the phrase does not affect the unit of work management.

The client application issues a commit or rollback statement for each transaction. When the client application closes its connection, the DB2 access service issues a rollback before exiting. Therefore, the client must commit any work that should be committed.

If the client application does not issue timely commits or rollbacks, then host resources, such as the APPC session or DB2 logging and locking, are held for an indeterminate amount of time. Therefore, long transactions can cause performance problems for other applications that need to access the same resource.

See the Managing processing section that follows for more information about the Allocate and StopCondition properties, and how they interact with the transaction mode.

Managing processing

You can control processing by using configuration properties to determine the following:

- How many rows are returned (MaxRowsReturned)
- Whether to stop when an error occurs (StopCondition)
- How to allocate conversations with a target DB2 system (Allocate)

For more information about the preceding properties, see Chapter 2, “Creating and Configuring DB2 Access Services.”

Note To set values for properties in the configuration file, you must know the specific category the property belongs to, verify that the category exists in the file, and enter the property value under the category.

MaxRowsReturned property

The MaxRowsReturned property specifies the maximum number of rows retrieved in a result set. (A result set is all or part of the results from a processed SQL statement.) However, one SQL request can produce multiple result sets.

Note If the number of rows exceeds the value of the MaxRowsReturned property, the DB2 access service returns the maximum number of rows and issues a warning message. However, the DB2 access service issues warning messages only when the SendWarningMessages property is enabled.

StopCondition property

The StopCondition property specifies whether the DB2 access service stops processing a request when it encounters an error or a warning.

Valid values are:

- error
- warning
- none

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

Allocate property

The Allocate property specifies when the APPC conversation that exists between the DB2 access service and the target database system is allocated and deallocated. Valid values are connect and request.

- If you specify connect (the default), the APPC conversation remains allocated until the client issues some form of deallocation, such as an exit. As a result, fewer APPC connections are available, but overhead for each client is reduced.
- If you specify request, resources are generally released sooner and, as a result, more clients can access the target database at a time. However, overhead increases because of the repeated initiation of APPC conversations for each request handled. You can specify request for either short or long transactions.

Effects of combined property settings on transaction behavior

The tables on the following pages show how combined configuration property settings can affect processing results. The first two tables are based on the TransactionMode property settings of short and long. The third table shows processing behavior when a begin transaction statement occurs.

Transaction mode = short

The following four tables show the effects of combined StopCondition and Allocate properties, based on a TransactionMode setting of short.

Allocate = request, StopCondition = error

The following table shows the effects of a TransactionMode property setting of short, an Allocate property of request, and a StopCondition property of error.

Table 5-1: Transaction mode = short, Allocate = request, StopCondition = error

Condition/Status	Effects of settings on processing
Does a rollback occur on an error?	The transaction rolls back immediately.
What is the status of the connection after an error occurs?	The connection ends.
What happens if a begin transaction occurs?	A commit occurs and uses begin transaction behavior described in Table 5-9.
When the client application issues a commit or rollback, what happens?	A commit or rollback occurs.
If the request is a cursor or dynamic request, how does it behave?	The request behaves the same as TransactionMode=long.
In batch mode, what happens if an error occurs?	The batch ends and a rollback occurs.
When completed, is a batch job committed?	The batch commits if no errors occur.
After the batch commits, what is the status of the connection?	The connection ends.

Allocate = request, StopCondition = none

The following table shows the effects of a TransactionMode property setting of short, an Allocate property of request, and a StopCondition property of none.

Table 5-2: TransactionMode = short, Allocate = request, StopCondition = none

Condition/Status	Effects of settings on processing
Does a rollback occur on an error?	A rollback does not occur.
What is the status of the connection after an error occurs?	The connection continues.
What happens if a begin transaction occurs?	A commit occurs and uses begin transaction behavior described in Table 5-9.
When the client application issues a commit or rollback, what happens?	A commit or rollback occurs.
If the request is a cursor or dynamic request, how does it behave?	The request behaves the same as TransactionMode=long.
In batch mode, what happens if an error occurs?	The batch continues executing.
When completed, is a batch job committed?	The batch is always committed.
After the batch commits, what is the status of the connection?	The connection ends.

Allocate = connect, StopCondition = error

The following table shows the effects of a TransactionMode property setting of short, an Allocate property of connect, and a StopCondition property of error.

Table 5-3: TransactionMode = short, Allocate = connect, StopCondition = error

Condition/Status	Effects of settings on processing
Does a rollback occur on an error?	The transaction rolls back immediately.
What is the status of the connection after an error occurs?	The connection continues.
What happens if a begin transaction occurs?	A commit occurs and uses begin transaction behavior described in Table 5-9.
When the client application issues a commit or rollback, what happens?	A commit or rollback occurs.
If the request is a cursor or dynamic request, how does it behave?	The request behaves the same as TransactionMode=long.
While in batch, what happens if an error occurs?	The batch request ends.
When completed, is a batch job committed?	The batch commits if no errors occur.
After the batch commits, what is the status of the connection?	The connection continues.

Allocate = connect, StopCondition = none

The following table shows the effects of a TransactionMode property setting of short, an Allocate property of connect, and a StopCondition of none.

Table 5-4: TransactionMode = short, Allocate = connect, StopCondition = none

Condition/Status	Effects of settings on processing
Does a rollback occur on an error?	A rollback does not occur.
What is the status of the connection after an error occurs?	The connection continues.
What happens if a begin transaction occurs?	A commit occurs and uses begin transaction behavior described in Table 5-9.
When the client application issues a commit or rollback, what happens?	A commit or rollback occurs.
If the request is a cursor or dynamic request, how does it behave?	The request behaves the same as TransactionMode=long.
In batch mode, what happens if an error occurs?	The batch continues.
When completed, is a batch job committed?	The batch commit occurs, if no errors occur.
After the batch commits, what is the status of the connection?	The connection continues.

Transaction mode = long

The following four tables show the effects of StopCondition and Allocate properties, based on a TransactionMode setting of long.

Allocate = request, StopCondition = error

The following table shows the effects of a TransactionMode property setting of long, an Allocate property of request, and a StopCondition of error.

Table 5-5: TransactionMode = long, Allocate = request, StopCondition = error

Condition/Status	Effect of settings on processing
Does a rollback occur on an error?	A rollback does not occur.
What is the status of the connection after an error occurs?	The connection continues.
What happens if a begin transaction occurs?	The begin transaction is ignored.
When the client application issues a commit or rollback, what happens?	If TransactionMode was originally short, and cursors and dynamics were all freed, the commit or rollback occurs and TransactionMode changes back to short. If TransactionMode was originally long, then a commit or rollback occurs.
If the request is a cursor or dynamic request, how does it behave?	If TransactionMode was originally short, and cursors and dynamics were all freed, a commit occurs and TransactionMode changes back to short. If TransactionMode was originally long, then the connection behaves the same as Allocate=request until all cursors and dynamics are freed.
In batch mode, what happens if an error occurs?	The batch transaction ends.
When completed, is a batch job committed?	The batch does not commit.
After the batch commits, what is the status of the connection?	The connection ends if the commit or rollback occurs at the end of batch. Otherwise, the connection continues.

Allocate = request, StopCondition = none

The following table shows the effects of a TransactionMode property setting of long, an Allocate property of request, and a StopCondition of none.

Table 5-6: TransactionMode = long, Allocate = request, StopCondition = none

Condition/Status	Effects of settings on processing
Does a rollback occur on an error?	A rollback does not occur.
What is the status of the connection after an error occurs?	The connection continues.
What happens if a begin transaction occurs?	The DB2 access service ignores the begin transaction command.
When the client application issues a commit or rollback, what happens?	If TransactionMode was originally short, and cursors and dynamics were all freed, the commit or rollback occurs and TransactionMode changes back to short. If TransactionMode was originally long, then a commit or rollback occurs.
If the request is a cursor or dynamic request, how does it behave?	If TransactionMode was originally short, and cursors and dynamics were all freed, a commit occurs and TransactionMode changes back to short. If TransactionMode was originally long, then the connection behaves the same as Allocate=request until all cursors and dynamics are freed.
While in batch, what happens if an error occurs?	The batch continues.
When completed, is a batch job committed?	The batch does not commit.
After the batch commits, what is the status of the connection?	The connection ends if the commit or rollback occurs at the end of the batch. Otherwise, the connection continues.

Allocate = connect. StopCondition = error

The following table shows the effects of a TransactionMode property setting of long, an Allocate property of connect, and a StopCondition of error.

Table 5-7: TransactionMode = long, Allocate = connect, StopCondition = error

Condition/Status	Effects of settings on processing
Does a rollback occur on an error?	A rollback does not occur.
What is the status of the connection after an error occurs?	The connection continues.
What happens if a begin transaction occurs?	The DB2 access service ignores the begin transaction command.
When the client application issues a commit or rollback, what happens?	If TransactionMode was originally short, and cursors and dynamics were all freed, the commit or rollback occurs and TransactionMode changes back to short. If TransactionMode was originally long, then a commit or rollback occurs.
If the request is a cursor or dynamic request, how does it behave?	If TransactionMode was originally short, and cursors and dynamics were all freed, the commit occurs and TransactionMode changes back to short. If TransactionMode was originally long, then the connection behaves the same as Allocate=request until all cursors and dynamics are freed.
While in batch, what happens if an error occurs?	The batch request ends.
When completed, is a batch job committed?	The batch does not commit.
After the batch commits, what is the status of the connection?	The connection continues.

Allocate = connect, StopCondition = none

The following table shows the effects of a TransactionMode property setting of long, an Allocate property of connect, and a StopCondition of none.

Table 5-8: TransactionMode = long, Allocate = connect, StopCondition = none

Condition/Status	Effects of settings on processing
Does a rollback occur on an error?	A rollback does not occur.
What is the status of the connection after an error occurs?	The connection continues.
What happens if a begin transaction occurs?	The DB2 access service ignores the begin transaction command.
When the client application issues a commit or rollback, what happens?	If TransactionMode was originally short, and cursors and dynamics were all freed, the commit or rollback occurs and TransactionMode changes back to short. If TransactionMode was originally long, then a commit or rollback occurs.
If the request is a cursor or dynamic request, how does it behave?	If TransactionMode was originally short, and cursors and dynamics were all freed, a commit occurs and TransactionMode changes back to short. If TransactionMode was originally long, then the connection behaves the same as Allocate=request until all cursors and dynamics are freed.
In batch mode, what happens if an error occurs?	The batch continues.
When completed, is a batch job committed?	The batch does not commit.
After the batch commits, what is the status of the connection?	The connection continues.

Effect of begin transaction command

The following table shows the effect on processing when a begin transaction command occurs in a request.

Table 5-9: Effects of begin transaction command on processing

Condition/Status	When begin transaction occurs...
Does a rollback occur on an error?	A rollback does not occur.
What is the status of the connection after an error occurs?	The connection continues.
What happens if another begin transaction occurs?	The DB2 access service ignores the begin transaction command.
When the client application issues a commit or rollback, what happens?	If TransactionMode was originally short, it changes back to short. If TransactionMode was originally long, then a commit or rollback occurs.
If the request is a cursor or dynamic request, how does it behave?	The request continues behaving as a short setting.
In batch mode, what happens if an error occurs?	The batch continues.
When completed, is a batch job committed?	The batch does not commit.
After the batch commits, what is the status of the connection?	The connection continues.

Troubleshooting

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

Configuration properties control whether data is recorded in the server log file and server trace file for each logging and tracing option. To configure log and trace properties, edit the DB2 Access Service Library configuration file or use DirectConnect Manager.

For detailed information about DB2 Access Service Library and DB2 access service logging and tracing properties, see Chapter 2, “Creating and Configuring DB2 Access Services” in this manual.

For information about server logging and tracing properties, see the DirectConnect *Server Administration Guide*.

Logging options

Log properties allow you to record information for DB2 access service administration. Logging options are controlled by the server, DB2 Access Service Library, and DB2 access service configuration properties. Each logging option requires the configuration property name and value.

Tracing options

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

Warning! To provide Sybase Technical Support with all necessary 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 DirectConnect *Server Administration Guide* for suggestions for deleting or backing up old log and trace files.

For information about configuration property syntax, see "Tracing properties" on page 61.

CHAPTER 6 Issuing SQL Statements

This chapter describes the SQL commands that you can issue, and how DirectConnect and the DB2 access service process these commands.

This chapter contains the following topics:

SQL transformation modes

Every database has its own dialect of SQL. Adaptive Server uses a dialect of SQL called Transact-SQL® (T-SQL). Both IBM and Sybase Adaptive Server SQL syntaxes support the ANSI SQL-1 standard. Thus, applications written for specific tables are relatively portable between Adaptive Server and DB2. However, the DB2 access service does not support all SQL statements. If unsupported extensions exist in a SQL statement, the DB2 access service passes the SQL statement to the target, which may accept it or return a syntax error.

To make the various dialects look like common SQL, the DB2 access service supports two transformation modes, called sybase and passthrough. The section called "Description of passthrough and sybase transformation modes" on page 108 describes these two modes in more detail.

Although the transformation mode primarily affects the way the DB2 access service treats incoming SQL statements, it also affects the following functional areas:

- Transaction management
- Datatype handling
- set statements
- Global variable processing

- DB2 stored procedures
- Catalog stored procedures (CSPs)
- System stored procedures
- Remote stored procedures (RSPs)
- Host-resident requests
- Interoperability with:
  - ASE/CIS (formerly OmniConnect™)
  - ODBC driver
  - Replication Server

When you configure the DB2 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.

Using the SQLTransformation property, you can specify how the DB2 access service processes SQL syntax. The SQLTransformation property is described on page 59.

The DB2 access service *never* transforms the following SQL statements:

- set statements
- select @@global variable statements
- transfer statements
- CSP requests

Instead, the DB2 access service processes these statements and takes any corresponding action needed.

Description of passthrough and sybase transformation modes

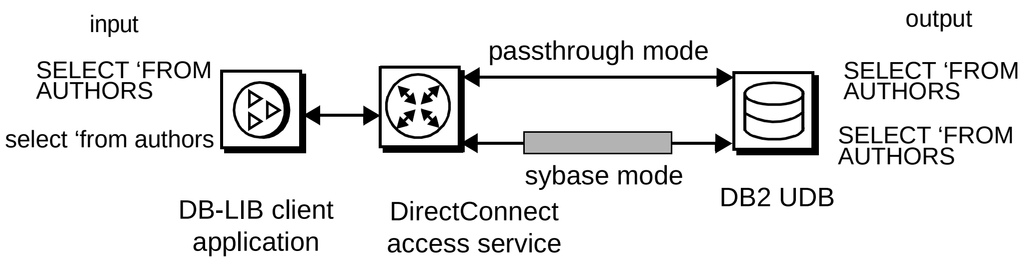
By translating dialect and syntax of the SQL statements the DB2 access service receives, passthrough and sybase transformation modes allow you to write applications that meet your needs. The following table shows you what to consider when choosing the transformation mode.

Table 6-1: Selection criteria for transformation mode

Use passthrough mode for...	Use sybase mode for...
Optimum performance	Portability across different database systems
Use of DB2-specific dialect features	Potential reuse of some existing applications written for Adaptive Server

As shown in the following figure, passthrough mode transfers similar dialect and syntax directly from the client application to the target database, saving time and resources. The sybase mode performs translation functions, changing the select statement from lowercase in the client application to uppercase in the target database.

Figure 6-1: Passthrough and sybase transformation modes



The following sections describe each transformation mode.

passthrough mode

Use passthrough mode when you want the client to have direct access to the capabilities of a DBMS target. In fact, the client must issue statements in the target SQL dialect, because the DB2 access service does not perform any SQL transformation. The passthrough transformation mode does convert carriage returns and control characters (such as line feeds) to blanks, unless they are part of a quoted string. Otherwise, SQL statements pass untouched to the DBMS, and the results of each are returned to the client application.

The passthrough mode does not add semicolons between statements, nor does it check statement syntax. These tasks are performed entirely by the target database. However, passthrough mode does convert all SQL keywords to uppercase when the DefaultTargetCodeset property value is other than 500 and 37.

Note For backward compatibility with the MDI Database Gateway, the DB2 access service changes any transformation mode request for DB2 or TSQL0 to passthrough mode, the default.

sybase mode

Use sybase mode for maximum compatibility between different target databases. This allows client applications that use sybase mode to potentially operate independently of the target they are accessing. CIS functionality in Adaptive Server and Replication Server require sybase mode when communicating with DirectConnect.

When sybase mode is in effect, the DB2 access service performs a limited amount of T-SQL syntax transformation on all the SQL text that it receives. This includes text found in the following commands:

- language
- cursor declare
- dynamic prepare
- dynamic execute immediate

In sybase mode, the DB2 access service transforms Sybase T-SQL it receives into syntax that the target DBMS supports. If the DB2 access service receives syntax that it does not recognize, it simply passes the text to the DBMS for execution. As a result, a single application should be able to generate somewhat complex queries that can apply consistently to all Sybase-provided DB2 access services.

Note Sybase mode supports multiple statements in a batch, with the exception of set and select@@.

Because an application uses this mode for purposes of compatibility with all DB2 access services provided by Sybase, it should not issue SQL commands that are unique to any single target DBMS.

Sybase mode makes the following transformations to SQL syntax:

- Transforms SQL commands shown in Table C-1 on page 237. Unsupported syntax passes unchanged to the database for processing.
- Transforms parameter marker names beginning with the @ character to a question mark (?).
- Removes T-SQL comments in the form /*.....*/.
- Converts T-SQL comparison operators (such as ! and =) into the target database equivalent.
- Converts nonquoted tokens to uppercase, when needed.
- Converts single and double quotation marks used as string delimiters to the appropriate delimiter for the target DBMS.
- Strips dollar signs from money constants.
- Adds a semicolon at the end of every statement.

Changing the transformation mode

To change the way the DB2 access service modifies SQL statements that are written for one database but are processed against another database, change the SQLTransformation property value.

Use one of the following methods to change the transformation mode:

- Issue a set statement to change the DB2 access service configuration until the connection is terminated or the mode is explicitly reset. For a description of set statements, see Chapter 3, "Querying and Setting Operating Values.".
- Change the DB2 access service configuration file and restart the DB2 access service. For information about how to change this file, see Chapter 2, "Creating and Configuring DB2 Access Services.".
- Create a new DB2 access service. For more information, see "How to create additional services" on page 16.

For a detailed list of supported SQL statements in DB2 and Adaptive Server, see Appendix C, "Using Sybase Mode Commands.".

Command types

As a developer, you can issue the following types of commands:

- Language commands
- Dynamic commands
- Cursor commands
- RPC requests

The first three topics are presented in this chapter. RPC requests are described in Chapter 7, "Issuing Remote Procedure Calls".

Language commands

Language commands are the simplest way to issue a request, but there are drawbacks to using them:

- You cannot control incoming datatype conversion.
- You must issue a request and finish processing it before you can issue another request.
- With certain commands, performance is slower than with dynamic and cursor commands.

Note that DirectConnect supports long character strings in language events for non-select statements: MainframeConnect for DB2 scans SQL statements for long character strings and replaces them with parameter markers. The long character string is then passed to DB2 with a SQL descriptor area (SQLDA).

Note Long character-string substitution is not done for select statements.

Dynamic commands

Dynamic commands are available only with Open Client CT-Library System 10 or later. This section describes how DirectConnect processes dynamic commands through DB2.

Dynamic commands allow an application to execute SQL statements, such as insert, update, and delete, that contain variables with values of that are determined at runtime, in the following manner:

1. An application prepares a dynamic SQL statement by associating a SQL statement that contains placeholders with an identifier.
2. The statement goes through SQL transformation, where it is precompiled. The results depend on the SQL transformation mode.
3. The statement is stored in the DB2 access service, where it waits for values to replace the placeholders. The statement is then called a "prepared statement."
4. Once the placeholders are replaced by values, the DB2 access service sends the statement to the target for execution.

Following is an example of a dynamic SQL command:

```
insert into table2 (col1, col2) values (SALES1, SALES2)
```

Following is the same command using parameter markers:

```
insert into table2 (col1, col2) values (?, ?)
```

The DB2 access service prepares the preceding statement. The statement executes and substitutes the values *SALES1* and *SALES2* into the parameter markers.

You can prepare and execute dynamic SQL statements by mapping these commands to DB2 capabilities. For example:

- You can prepare any non-select statement and execute the prepared statement as many times as you want.
- You can execute non-select statements through the dynamic command execute immediate capability.

For more information about how to prepare and execute statements with dynamic SQL, see the Open Client *Client-Library/C Reference Manual* and the *IBM DB2 SQL Reference Manual*.

Capabilities and limitations of dynamic commands

Following is a description of dynamic command capabilities that DirectConnect supports, as well as the limitations that apply.

- You can only prepare use procedure, transfer, and most non-select SQL statements, such as insert, update, and delete statements. For a current list of SQL statements, see the *IBM DB2 SQL Reference* manual.
- The DB2 access service does not support the from clause in an update statement.
- The DB2 access service supports a maximum of 50 simultaneously-prepared dynamic statements per client connection.
- releases all resources associated with the prepared statement. The client application must also free its connection at this point so that CT-Library synchronizes with the actual state of both DB2 and the DB2 access service.
- The DB2 access service can support both short and long transactions in either allocate on request or allocate on connect mode.
- The first time an application issues a prepare (or declares a cursor), the DB2 access service forces TransactionMode for that connection to long. The Allocate property remains unchanged.
- The DB2 access service supports the execute immediate capability of a dynamic command, but the statement must not return data. (This is an Open Server restriction, not a DB2 access service limitation.) The DB2 access service allows a use procedure statement (as long as the stored procedure does not return data) but not a transfer statement.
- Parameter markers for dynamic commands are supported as follows:
    - In sybase mode, parameter markers within the text can use Sybase conventions, such as @paramname. When necessary, the DB2 access service transforms parameters to the native syntax.

    - In passthrough mode, parameter marker syntax is database-specific, so you must use question marks (?) as parameter markers.
- The DB2 access service also supports long character and binary parameters. Client applications must describe such properties as datatype CS_LONGCHAR or CS_LONGBINARY.

  Consider the following restrictions:

    - DB2 limits the SQL statement for any request (such as a prepare) to a total of 32,765 bytes. The maximum size of both data and null indicators in a row is 32,767 bytes.
    - Open Server Tabular Data Stream (TDS) allows a maximum TextSize property of 32,000 bytes.

For general information about dynamic SQL and dynamic commands, see the Sybase Open Client *Embedded SQL Programmer's Guide* and the *IBM DB2 SQL Reference* manual.

Note The DB2 access service does not support Embedded SQL (ESQL). However, ASE/CIS does support Embedded SQL. You could configure ASE/CIS before DirectConnect to support this functionality.

Using dynamic commands

When you write an application using dynamic commands, consider the following:

- The SQL statement being prepared must not be a select statement. Although Open Server allows a select statement to be prepared and a cursor to be opened on the prepared statement, the DB2 access service does not support this capability.
- Use DB2 syntax for the statement you are preparing when you are in passthrough transformation mode.

Following is an example of a code fragment that prepares and executes an insert statement.

```
/*
 ** This sample code fragment illustrates the CT-Library calls
 ** used to prepare and execute a SQL statement. Error handling
 ** and other details are omitted for the sake of brevity.
 ** NOTE: Ellipses in the following represent code that you must supply.
 */
```

```
/*
** Prepare the statement.  The statement we will prepare is:
** INSERT INTO TEST VALUES (?, ?, ?, ?).
** We will name the dynamic statement DYN1.
*/
retcode = ct_dynamic( cmd, CS_PREPARE, "DYN1", CS_NULLTERM,
   "INSERT INTO TEST VALUES (?, ?, ?, ?)", CS_NULLTERM );
/*
** Send the batch and check results.
*/
retcode = ct_send( cmd );
retcode = handleresults( cmd );
/*
** Now execute the prepared statement with a set of parameter
** values. Allocate a CS_DATAFMT structure for each parameter.
*/
dfmt = malloc( 4 * sizeof(struct CS_DATAFMT) );
/*
** Fill in the structure and set the datalength, null indicator,
** and data value for each parameter.
*/
dfmt[0].datatype = CS_CHAR_TYPE;
dfmt[0].status = CS_INPUTVAL;
dfmt[0].maxlength = strlen( "col1val" );
dataptr[0] = "col1val";
datalen[0] = strlen( "col1val" );
nullind[0] = 0;
...
/*
** Execute the statement.
*/
retcode = ct_dynamic( cmd, CS_EXECUTE, "DYN1", CS_NULLTERM,
   NULL, CS_UNUSED );
/*
** Describe and send the parameters.
*/
for (i=0; i<4; i++)
{
   retcode = ct_param( cmd, &dfmt[i], dataptr[0], datalen[i],
      nullind[i] );
}
/*
** Send the batch and check results.
*/
retcode = ct_send( cmd );
retcode = handleresults( cmd );
```

For more information about SQL processing, see the *IBM DB2 SQL Reference manual*.

Error handling

Special error handling is not required.

Cursor commands

Cursor commands, which are available only with Open Client CT-Library System 10 or later, give an application the power to retrieve and change data in a flexible way. For example, you can use cursor commands to process multiple result sets that are simultaneously available to the application, instead of one at a time using the language command.

Cursor commands require you to specify a SQL select statement that goes through SQL transformation, as follows:

1. The client application declares and opens the cursor command through the CT-Library routine ct_cursor.
2. When the cursor opens, a set of rows on the target is qualified. At that point, the select, update, or delete statement can initiate the following SQL statements to operate on specific rows in the set:
   - fetch
   - update
   - delete

Note The DB2 access service does not accept language event-based cursor commands. You must use ct_cursor commands.

DB-Library also supports a form of cursors but somewhat differently. The following table shows the differences between DB-Library cursors and CT-Library cursors.

Table 6-2: Comparison of DB-Library and CT-Library cursors

Characteristic	DB-Library-based cursor	CT-Library-based cursor
Relationship to Sybase SQL cursor	Called an “emulated” or “client-side” cursor, does not correspond to a SQL cursor.	Called a “native” or “server-side” cursor, corresponds to an actual SQL cursor.

Characteristic	DB-Library-based cursor	CT-Library-based cursor
Cursor row position	Defined by the client.	Defined by the server.
Fetch capability	Can fetch backward.	Can fetch forward only.
Memory requirements	More is required if querying large row sizes, unless a smaller number of rows in the buffer is specified.	No additional memory is required, regardless of the row sizes.
Access to Open Server application	Not available, unless required DB-Library stored procedures are installed.	Any System 10 or later Open Server application is coded to support cursors.

The DB2 access service supports CT-Library-based cursors by mapping the corresponding Open Server commands to DB2 capabilities. Because DB2 cursors do not provide all the capabilities of Open Server cursors, some limitations apply and are included in the following section.

For more information about DB2 capabilities related to commit and rollback statements, see the *IBM DB2 SQL Reference* manual.

Capabilities and limitations of cursor commands

Following is a description of cursor capabilities that DirectConnect supports, as well as the limitations that apply:

- Only CT-Library client applications built on Open Client System 10 or later can use cursor capabilities.
- The DB2 access service supports a maximum of 50 simultaneously declared cursors for each client connection.
- In both sybase and passthrough modes, the DB2 access service does not send text with the cursor delete command; instead, it formulates the DBMS-specific equivalent of:

  delete from <*tablename*> where current of cursor <*cursorname*>

  where <*cursorname*> is based on the current Open Server cursor ID.
- The DB2 access service does not support the from clause in an update statement.
- When the client application issues a rollback, DB2 closes and frees all cursors. The client application must also close and free its cursors at this point, so that CT-Library synchronizes with the actual state of the cursors.
- The client application can select the behavior of cursors after a commit; in other words, the cursor is either closed, or it is not closed and retains its position:

    - If the client executes a set CloseOnEndtran on command before any cursors were declared, then all cursors are closed—but not freed—after the commit. The client application closes its cursors so that CT-Library synchronizes with the actual state of the cursors. If the application does not execute the close on endtran statement, on is the default, so the statement executes anyway.
    - If the client executes a set CloseOnEndtran off command before any cursors are declared, then all cursors remain open and retain their current position after a commit occurs. This corresponds to the with hold clause in DB2.
- A parameter marker can vary depending on the transformation mode:
    - In passthrough mode, the marker is a question mark (?).
    - In sybase mode, the marker is an (@) “at”sign.
- The DB2 access service supports long character and binary parameters. Client applications must describe such parameters as CS_LONGCHAR or CS_LONGBINARY.

Note DB2 limits the SQL statement for any request (such as a cursor declare or update) to 32,765 bytes. With data and null indicators added, the total limit is 32,767.

- The DB2 access service does not allow you to declare a cursor in a dynamically prepared select statement.

Using read-only cursors

When you write an application using read-only cursors, consider the following:

- The DB2 access service supports forward positioning (next) only. It does not support any other fetch option, such as previous, first, last, absolute, or relative.
- You must specify the cursor as read_only when declaring it.
- The read-only cursors are set for fetch only, so rows cannot be deleted.

The fetch count is automatically set to 1, unless you set it to a different number. If you change the fetch count to 20, DirectConnect fetches 20 rows at a time.

The following code fragment shows the sequence of CT-Library calls used to declare, open, fetch, close, and free a read-only cursor.

```
/*
 ** This sample code fragment illustrates the CT-Library calls that are
 ** used to declare, open, fetch, close and deallocate a read-only
 ** cursor. Error handling and other details are omitted for the sake of
 ** brevity.
 ** NOTE: Ellipses in the following represent code that you must supply.
 */
       /*
       ** Initialize CT-Lib, establish a connection to DirectConnect.
       */
       ...
       /*
       ** Set cursor behavior on COMMIT so that cursors will maintain
       ** their position and not be closed.
       */
       retcode = ct_cmd_alloc( connection, &cmd );
       strcpy( lang, "SET CLOSEONENDTRAN OFF" );
       retcode = ct_command( cmd, CS_LANG_CMD, lang, CS_NULLTERM,
          CS_UNUSED );
       /*
       ** Send the batch and check results.
       */
       retcode = ct_send( cmd );
       retcode = handleresults( cmd );
       /*
       ** We are going to declare the cursor on the statement:
       **      SELECT * FROM AUTHORS WHERE STATE = ?
       **
       ** Allocate a CS_DATAFORMAT structure for the parameter and fill
       ** in the relevant fields. In this case, the parameter is char(2).
       */
       dfmt = malloc( sizeof(CS_DATAFMT) );
       memset( dfmt, 0, sizeof(CS_DATAFMT) );
       dfmt->status = CS_INPUTVALUE;
       dfmt->format = CS_UNUSED;
       dfmt->datatype = CS_CHAR_TYPE;
       /*
       ** Declare the cursor, using cursor name "CURS1".
       */
       retcode = ct_cursor( cmd, CS_CURSOR_DECLARE, "CURS1", CS_NULLTERM,
          "SELECT * FROM AUTHORS WHERE STATE = ?", CS_NULLTERM,
          CS_READ_ONLY );
       /*
       ** Describe the parameter.
       */
       retcode = ct_param( cmd, &dfmt, NULL, CS_UNUSED, 0 );
```

```
/*
** Set the fetch count to 20.
*/
retcode = ct_cursor( cmd, CS_CURSOR_ROWS, NULL, CS_UNUSED,
   NULL, CS_UNUSED, 20 );
/*
** Send the batch and check results.
*/
retcode = ct_send( cmd );
retcode = handleresults( cmd );
/*
** Now open the cursor with host variable value = 'CA'
*/
strcpy( parmval, "CA" );
datlen = 2;
nullind = 0;
retcode = ct_cursor( cmd, CS_CURSOR_OPEN, NULL, CS_UNUSED, NULL,
   CS_UNUSED, CS_UNUSED );
/*
** Send the parameter.
*/
retcode = ct_param( cmd, &dfmt, parmval, datlen, nullind );
/*
** Send the batch and check results.
*/
retcode = ct_send( cmd );
retcode = handleresults( cmd );
/*
** Describe and bind the result set.
** Find out how many columns there are in this result set.
** Make sure that there is at least 1 column.
*/
retcode = ct_res_info( cmd, CS_NUMDATA, &ncols, CS_UNUSED, NULL );
/*
** Allocate memory for each column's CS_DATAFMT, a data pointer,
** data length, and null indicator.
*/
...
/*
** Loop through the columns getting a description of each one and
** binding each one to a program variable.
*/
for (i = 0; i < ncols; i++)
{
   /*
   ** Get the column description. ct_describe() fills the
```

```
        ** datafmt parameter with a description of the column.
        */
        retcode = ct_describe( cmd, (i + 1), &(coldata[i].dfmt) );
        /*
        ** Update the datafmt structure to indicate the form in
        ** which we want to get the data. Set the datatype,
        ** format, maximum length, etc.
        */
        ...
        /*
        ** Allocate memory for the actual column data.
        */
        ...
        /*
        ** Now bind.
        */
        retcode = ct_bind( cmd, (i + 1), &(coldata[i].dfmt ),
        coldata[i].data, &( coldata[i].datlen ),&( coldata
        [i].nullind) );
     }
     /*
     ** Fetch 1 row.
     */
     retcode = ct_fetch( cmd, CS_UNUSED, CS_UNUSED, CS_UNUSED,
        &rows_read );
     /*
     ** Check the results of the fetch.
     */
     if (retcode == CS_ROW_FAIL)
     {
     /*
     ** Fetch failed.
     */
     ...
     }
     else if (retcode != CS_SUCCEED)
{
     if (retcode == CS_END_DATA)
     {
        /*
        ** End of data has been reached.
        */
        retcode = handleresults( cmd );
        goto CLOSECURS;
     }
     else
```

```
      {
         /*
         ** Fetch failed.
         */
         ...
      }
}
/*
** We have a row of data.
*/
...
CLOSECURS:
/*
** Close and free the cursor.
*/
retcode = ct_cursor( cmd, CS_CURSOR_CLOSE, NULL, CS_UNUSED, NULL,
CS_UNUSED, CS_DEALLOC );
/*
** Send the batch and check results.
*/
retcode = ct_send( cmd );
retcode = handleresults( cmd );
```

Using updatable cursors

When you write an application using updatable cursors, consider the following:

- The DB2 access service supports only forward positioning with the fetch next option and with a fetch count of 1. It does not support any other fetch option (previous, absolute, relative). Therefore, it deletes or updates only the most recently fetched row.
- The SQL statement used for the cursor declaration must contain a for update of <*column_list*> clause.
- You must specify the cursor as for_update when you declare it.
- Use DB2 syntax for the statement being declared, which is determined by the transformation mode. When in sybase mode, the DB2 access service performs translation on the statement. Parameter markers can represent values that the application describes and sends to the DB2 access service. The DB2 access service then uses the parameter descriptions and values to create the final form of the update statement. Parameter markers are as follows:
  - In sybase mode, the marker is the (@) ”at” sign.

- In passthrough mode, the marker is a question mark (?).

- In both modes, the DB2 access service appends the where current of <*cursor_name*> clause to the update statement.

Following is a code fragment that shows the sequence of CT-Library calls used to delete and update a row on an updatable cursor.

```
/*
 ** This sample code fragment illustrates the CT-Library calls
 ** used to update and delete a row on an updatable cursor.
 **
 ** Error handling and other details are omitted for the sake of brevity.
 ** NOTE: Ellipses in the following represent code that you must supply.
 */
       /*
       ** See the previous example for the cursor declare, open and
       ** fetch. The differences are:
       ** 1. The statement to declare is
       ** SELECT * FROM AUTHORS WHERE STATE = ? FOR UPDATE OF PHONE
       ** 2. The declare must specify CS_FOR_UPDATE instead of
          CS_READ_ONLY.
       ** 3. The setting of the fetch count is eliminated.
       */
       /*
       ** Assume that you have fetched rows (one at a time) until you see
       ** a row you want to delete. This code deletes the most recently
       ** fetched row.
       */
       retcode = ct_cursor( cmd, CS_CURSOR_DELETE, "AUTHORS",
          CS_NULLTERM, NULL,CS_UNUSED, CS_UNUSED );
       /*
       ** Send the batch and check results.
       */
       retcode = ct_send( cmd );
       retcode = handleresults( cmd );
       /*
       ** Fetch more rows until we see a row that we want to update.
       */
       ...
       /*
       ** Update the most recently-fetched row.  For this example, we
       ** will update the PHONE column in the AUTHORS table, and we
       ** will use a parameter in the update statement for the value of
       ** PHONE. Fill in the CS_DATAFMT structure for the parameter and
       ** the parmval, atlen, and nullind values.
       */
       dfmt.status = CS_INPUTVALUE;
dfmt.datatype = CS_CHAR_TYPE;
strcpy( parmval, "303-443-2706" );
datlen = strlen( "303-443-2706" );
nullind = 0;
/*
** Send the cursor update.
*/
retcode = ct_cursor( cmd, CS_CURSOR_UPDATE, "AUTHORS",
   CS_NULLTERM,"UPDATE AUTHORS SET PHONE = ?", CS_NULLTERM,
   CS_UNUSED );
/*
** Send the parameter description and value.
*/
retcode = ct_param( cmd, &dfmt, parmval, datlen, nullind );
/*
** Send the batch and check results.
*/
retcode = ct_send( cmd );
retcode = handleresults( cmd );
```

Error handling for cursor declare and open commands

In general, the client application handles errors the same as any other CT-Library application. However, handling errors from a cursor declare or cursor open requires special consideration, as described next.

Cursor declare

If a cursor declare fails because, for example, a table does not exist or the user does not have sufficient privilege to access a table, the application should issue a ct_cancel and drop the command. This cleans up the necessary structures in Open Client and Open Server and synchronizes the client with both the DB2 access service and DB2.

Cursor open

If a cursor open fails because, for example, the number of parameters described in the open does not match the number of parameter markers in the declare, the client application should close and free the cursor and then drop the command. The client application should not issue a ct_cancel in this case.

Warning! If a cursor declare contains parameter markers but the cursor open does not describe any parameters, DirectConnect goes into an error state. As a result, the client's connection to the database is dropped when the next command is executed. To recover, the client application must close the connection to the DB2 access service and open a new one.

For more information about error handling, see the Sybase Open Client *Client-Library/C Reference Manual*.

CHAPTER 7 Issuing Remote Procedure Calls

This chapter describes how any CT-Library application can generate remote procedure call (RPC) events. The RPC feature allows a stored procedure to initiate an event in a remote database—that is, to make a remote procedure call.

A client application can invoke an RPC to:

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

This chapter contains the following topics:

Note When Adaptive Server issues an RPC event to the DirectConnect DB2 access service, it tries to establish a connection using the same service name as the identifier for the Adaptive Server in the *sql.ini* file (Windows), or the *interfaces* file (UNIX).

Setting up Adaptive Server and DirectConnect connections

❖ **To set up Adaptive Server for RPCs against the DirectConnect access service.**

1. Create an Adaptive Server user ID and password that matches a DirectConnect service host user ID and password.

   For instructions on creating the ID and password, see your Adaptive Server documentation.

2. Configure Adaptive Server for remote access.
3. Define each DirectConnect service as a remote server.
4. Define connectivity between Adaptive Sever and the DirectConnect DB2 access service.

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

Creating a SQL Server stored procedure

The following example shows a SQL Server stored procedure that can execute a remote stored procedure (RSP). The SQL Server stored 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 variable representing the customer name to be added.
- *@custno* is the variable representing the customer number to be added.
- servername specifies the DB2 access service instance to use. The three periods (. . .) following the *servername* are required.
- *addcust* is the stored procedure name on the host.

- @*addname* is the stored procedure variable representing the new customer name.
- @*addno* is the stored procedure variable representing the new customer number.

Configuring Adaptive Server for remote access

To configure Adaptive Server for remote access, log in to Adaptive Server as an administrator 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`, perform the following steps:

  a Enter:

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

  b Restart Adaptive Server.

Defining the DirectConnect service as a remote server

To define the DirectConnect DB2 access service as a remote server, enter each DirectConnect DB2 access service name in the Adaptive Server SYSSERVERS table using the following command:

sp_addserver *service_name*

where *service_name* is the name of the DirectConnect service you want to set up as a remote server. The name is case sensitive and must match the name you used to define the connectivity between Adaptive Server and the DirectConnect DB2 access service.

To verify that the DirectConnect service is successfully defined as a remote server, enter:

select *service_name* from sysservers

If data rows return, you successfully defined the DirectConnect access service as a remote server.

When Adaptive Server issues an RPC to a DirectConnect DB2 access service, it attempts to connect using a service name identical to the Adaptive Server identifier in the *interfaces* file.

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

- Define a service within the DB2 Access Service Library using the same name as Adaptive Server.
- Set up service name redirection to redirect the Adaptive Server to the appropriate service.

Executing a SQL Server stored procedure

To execute the preceding Adaptive Server stored procedure example using isql, connect to Adaptive Server and enter the following at the prompts:

```
C:>ISQL -Ssybone -Uuser -Ppasswrd
 1> execute newcust xxxx,yyyy
 2> go
```

where:

- *xxxx* is the new customer name, such as Ajax Printing Company.
- *yyyy* is the new customer number, such as 1234.

Note For backward compatibility, the following syntax is acceptable:
use procedure newcust *xxxx, yyyy*

Executing a language statement as an RPC

To execute a SQL language statement to the DB2 access service through a SQL Server RPC, use the following syntax:

```
C:> isql -Ssqlserver -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 special name, or keyword, of the RPC.

The DB2 access service RPC event handler is sensitive to several key RPC names. In this case, the RPC keyword dcon is a special name that DirectConnect recognizes. As a result, the DB2 access service translates the first parameter into a dynamic SQL statement, submits it to the target database, and then 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.

To execute a language statement as an RPC:

1. Adaptive Server determines if the remote server (in this case, directconnect) is configured as a remote server to Adaptive Server.
   - If the remote server is not configured, the request fails immediately.
   - If the remote server is configured, Adaptive Server checks for a site handler connection to the remote server.
2. Adaptive Server does one of the following:
   - If a site handler connection exists, Adaptive server connects to the remote server, triggering a connect event at the DirectConnect. If the connect event processes successfully, Adaptive Server triggers an RPC event at DirectConnect and submits the RPC dcon. The first parameter to the RPC is the dynamic SQL language statement that is executed.
   - If a site handler connection does not exist, Adaptive Server establishes one. This connection exists for the life of the RPC and is reused when 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. It connects to the remote server (DirectConnect) and submits the RPC. After results process, it disconnects.

These quick connects and disconnects are a basic part of the Adaptive Server RPC design to provide minimal network traffic.

Validation

The user ID and password you use to sign on to Adaptive Server *must* be a valid DirectConnect target database user ID and password.

Transformation mode and syntax

All SQL transformation rules apply:

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

Commitment control

Consider the following:

- If DirectConnect is configured for long transactions, the SQL submitted must not be sensitive to a commit. In other words, 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 DB2 access service is configured for short transactions, the SQL submitted is bound by short transactions, which supply a commit by default.

Creating a transfer SQL request

The client application can create a stored procedure within Adaptive Server that invokes the DB2 access service transfer function. In this case, the DB2 access service receives the transfer command as an RPC event with the arguments shown in the following table:

Table 7-1: Arguments for transfer command

Argument	Definition
Argument 1	The secondary connection information ({to | from} "server userid pw")
Argument 2	Either the bulk copy target command (for example, with replace into) or the destination-template sourceselectstatement
Argument 3	Either the bulk copy sourceselectstatement or the destination-template sourceselectstatement

Following is an example of a Adaptive Server stored procedure that initiates a bulk copy transfer statement. The DB2 access service executes the transfer statement against DB2, and DB2 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 DB2 access service that handles the transfer. The three periods (. . .) following the servername are required. The DB2 access service recognizes anything other than *transfer* in the next position as the name of a stored procedure.
- *transfer* is the name of the RPC.
- *servername2* is the secondary database for the transfer command.

Executing a transfer SQL request

As shown in the following example, to execute the *replauth* stored procedure using isql, connect to the Adaptive Server database and enter the following at the isql prompts:

```
C:>ISQL -Ssybone -Uuser -Ppasswrd
1> execute replauth
 2> go
```

When Adaptive Server executes *replauth*, it passes an RPC to the DB2 access service. The DB2 access service then returns any result rows or messages to the client application, not to Adaptive Server.

For more information about transfers, see Chapter 8, “Understanding the Transfer Process”.

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, which, when the phone column is updated in the authors table in Adaptive server, automatically calls an RSP named pchowdy. In turn, pchowdy updates the authors table on DB2, using au_id to specify the row to update.

```
create trigger updphone 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...pchowdy @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, the updphone trigger starts up whenever phone is updated:

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

Note The DB2 access service does not support a two-phase commit. The update can succeed and be committed on either platform, independently of the success or failure of the update on the other platform.

In the previous example, if the DB2 update fails, the DB2 access service rolls back the Adaptive Server transaction and shows the following message:

```
@ERR >> 0
```

For more information about RPCs, see the *Sybase Open Server Server-Library/C Reference Manual* or the *Sybase Open Client Client-Library/C Reference Manual*.

CHAPTER 8 Understanding the Transfer Process

This chapter describes several concepts of the transfer process.

This chapter 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 two transfer options: bulk copy and destination-template.

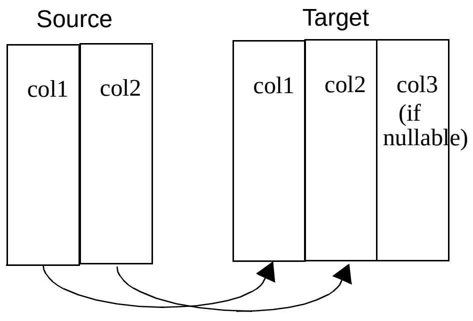
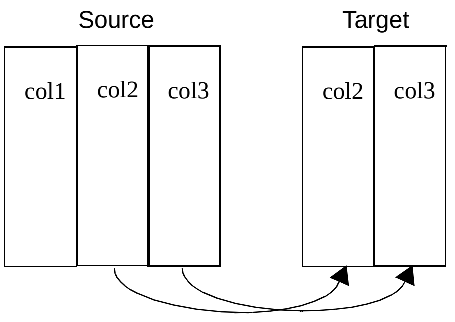
Bulk copy and destination-template transfer

Bulk copy transfer and destination-template transfer processes differ as follows:

- The bulk copy transfer process allows you to quickly and directly transfer large amounts of data that is compatible between the databases.
- The destination-template transfer process allows you the flexibility and control to perform some action on the data before sending it to the target database, by inserting it in a template.

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 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:  Source Target col1 col2 col1 col2 col3 (if nullable) transfer to secondary_connection; with replace into target_table; select col1, col 2 from source_table	Perform arbitrary actions against a target database using input from the source database, such as moving tables when the source table has more columns than the target:  transfer to secondary_connection; select col2, col3 from source_table; insert into target_table values (?,?)

Note In both types of transfer, the transfer statement must be the *only* statement in a request.

For detailed information about bulk copy transfer, see Chapter 9, "Using Bulk Copy Transfer." For detailed information about destination-template transfer, see Chapter 10, "Using Destination-Template Transfer."

Description of terms in the transfer process

Following are terms used in the transfer process discussion.

Primary database

This refers to the DBMS in the transfer statement. For DB2 access services, this is a DB2 database.

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 DB2 access service. *Always* specify the secondary database in transfer statements.

Source database

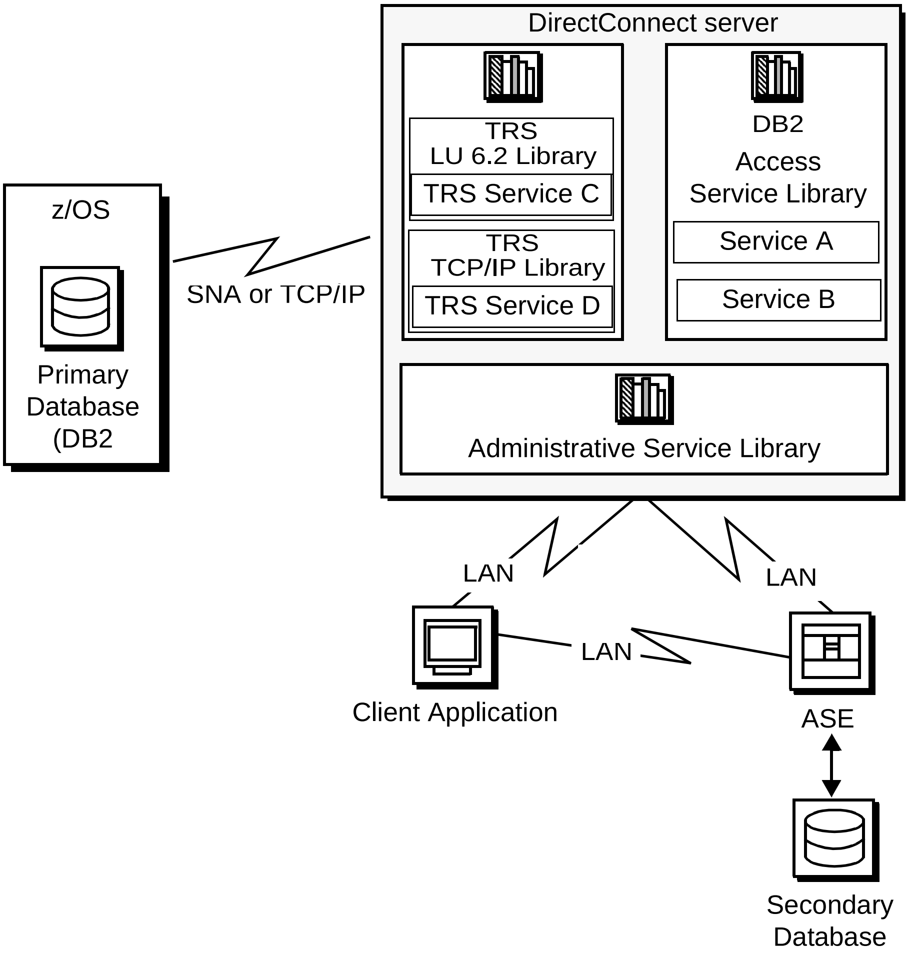
The source is the database the data is coming from.

Target 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 target, depending on whether you use the transfer from or transfer to command.

The following figure shows the DB2 access service configured to access a particular DB2 database, which is the primary database. The secondary database is an Adaptive Server database located on the LAN. The client application can transfer data from DB2 to ASE, or from Adaptive Server to DB2.

Figure 8-1: DB2 access service data transfer between databases



During a transfer:

- Data flows from a table in the source database, through the DB2 access service, to the target database. Although the client application initiates the transfer, the data does *not* flow through it.
- The DB2 access service becomes a client to the secondary database.

Transfer direction

You can transfer data in two directions:

- A transfer to statement transfers data from the *primary* database, DB2. This means that 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. This means that the secondary database becomes the source database to the primary database, DB2, 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, DB2, becomes the destination database, or target.

Note For implications of using one transfer direction over another, see “Datatype conversion for transfer processing” on page 143.

Unit of work

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 and destination-template transfer.

Bulk copy 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 DB2 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 DB2 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 145.

Destination-template transfer

When a destination-template transfer statement moves data from Adaptive Server to DB2, the DB2 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 DB2 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 DB2 access service issues a commit at the end of the entire transfer. Because StopCondition is set to none, the DB2 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 DB2 and:

- Adaptive Server
- ASE/CIS (formerly OmniConnect)
- Other DB2 access service and legacy products or services:
  - DB2 UDB
  - Informix
  - Microsoft SQL Server
  - AS/400
  - Any other database supported
- Any Open Server application that supports ASE

Datatype conversion for transfer processing

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

Datatype conversion is handled differently for the two transfer types. After converting the incoming source database datatypes into appropriate Open Client and Open Server datatypes, the DB2 access service does one of the following:

- For bulk copy transfer processing, the DB2 access service converts Open Client and Open Server datatypes 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 DB2 access service uses the datatype qualifiers specified with the question marks in the template. When the questions marks do not have qualifiers, the DB2 access service uses the datatypes of the source to determine the default qualifiers.

When conversion fails, several properties, such as DefaultDate, NumConvertError, and CharConvertError, fill the field with the default value.

Note When you transfer data between two DirectConnect DB2 access services (such as DB2 to AS/400, AS/400 to DB2), Sybase recommends that you execute the transfer from command from DB2 to guarantee native datatype mapping. Also, the transfer from command is the only way you can transfer text and image columns between DB2 access services.

Transfer errors and error handling

Transfer processing errors can occur due to:

- Structural errors, for which the DB2 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

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 transfers, before the DB2 access service moves any data from the source database to the target database, the DB2 access service compares datatypes in the source to datatypes in the target.
- In destination-template transfers, the DB2 access service compares the source table datatypes to the qualifiers set in the destination-template statement.

For bulk copy and destination-template transfer types, if any columns have incompatible datatypes, the DB2 access service cancels the transfer, without attempting to transfer any rows.

Incompatible number of columns

When the DB2 access service detects an unequal number of columns between the source and the target, a structural error can occur.

- For bulk copy transfer:
  - If the number of source columns exceeds the number of target columns, the DB2 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 DB2 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 DB2 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 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.

The DB2 access service handles these errors using the following properties:

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

If the SendWarningMessages property is set to yes, the DB2 access service sends a message to the client application when it encounters value errors.

In addition, as mentioned in "Datatype conversion for transfer processing" on page 143, the preceding properties can be used to fill in default values when datatype conversion fails during both types of transfer.

For more information about properties, see Chapter 2, "Creating and Configuring DB2 Access Services."

For information about values that cause errors for DB2 access service during bulk copy transfer, see "Bulk copy transfer errors" on page 155.

Error reporting for transfer processing

You can use one of three methods to obtain error information about transfer processing:

- Include the with report phrase in the transfer statement. When you include this phrase, the DB2 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 after 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 DB2 access service returns warning messages to the client when data conversion errors occur.

TransferErrorCount property

During transfer processing, the DB2 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 DB2 access service to ignore errors.

CHAPTER 9

Using Bulk Copy Transfer

This chapter describes the bulk copy transfer statement and contains the following topics:

Overview

Bulk copy transfer initiates a direct transfer of data between two databases from the client application. You use the bulk copy transfer statement to copy large amounts of data between similar tables. The following limitations apply.

Limitations for bulk copy transfer

- 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 transfer to work, the secondaryname to the secondary database must be recorded in the *interfaces* file for UNIX, or in the *sql.ini* file for Windows.
- 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 steps describe how bulk copy transfer processes data between two databases:

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 these criteria:

   - The target table must have at least as many columns as the result set.
   - The datatype of each result set column must be convertible to the type of its 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 order in which they appear. The access service attempts to insert NULL values (if allowable) in all columns in the target table that do 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 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

8 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 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 *interfaces* file for UNIX, or in the *sql.ini* file for Windows.
  - 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.

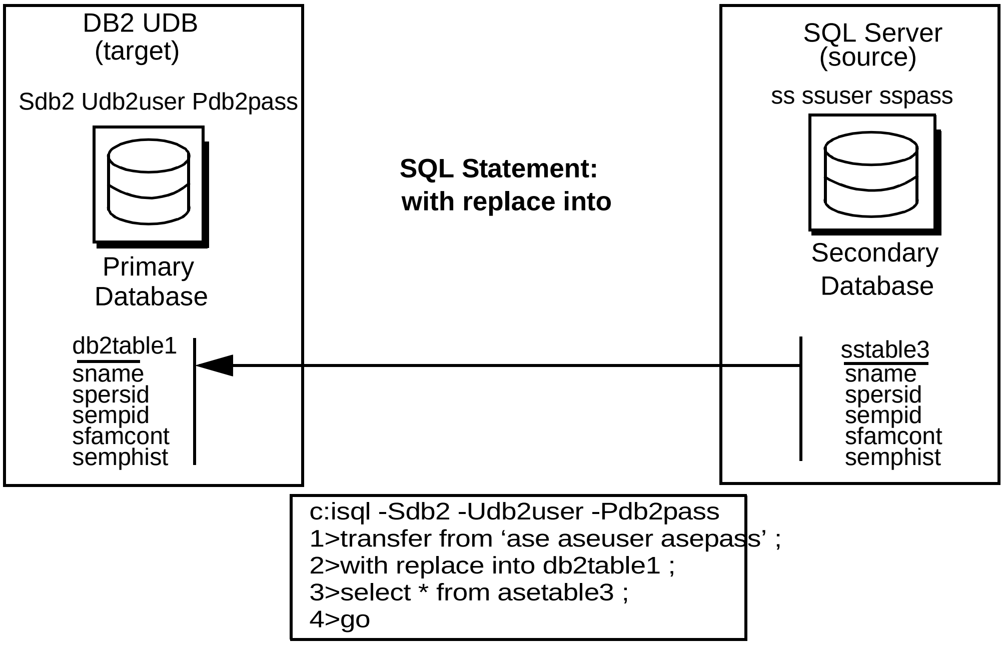
  This statement can be any statement the source database will accept, including stored procedures. SQL transformation is not performed on the sourceselectstatement. It must be in the source database SQL dialect.

Processing bulk copy “transfer from” statements

Use transfer from when you want to transfer data *from* a secondary database, either Adaptive Server or another database through another DB2 access service, to the primary database. The primary database, DB2, is the target and is implied in the transfer statement; the secondary and source database is specified in the transfer statement.

The following figure shows the data flow of one of the preceding transfer from examples, which transfers data from sstable3 in Adaptive Server to replace data in the DB2 table named db2table1.

Figure 9-1: DB2 bulk copy "transfer from" statement



```
c:isql -Sdb2 -Udb2user -Pdb2pass
1>transfer from 'ase aseuser asepass' ;
2>with replace into db2table1 ;
3>select * from asetable3 ;
4>go
```

Datatype conversion for "transfer from" processing

The DB2 access service performs datatype conversions for bulk copy transfer from statements. These conversions are defined within the DB2 access service and are not affected by the settings of the datatype conversion configuration properties and set statements.

Each DB2 access service specification will include specific definitions of datatype mapping for that target DBMS. The DB2 access service converts these datatypes into the actual datatypes of the target columns. If the DB2 access service cannot convert the columns, the DB2 access service ends the transfer with an error.

The DB2 access service performs datatype conversions for bulk copy transfers. These conversions are defined within the DB2 access service and are not affected by the settings of the datatype conversions configuration properties. Each DB2 access service specification includes specific definitions of datatype mapping for that target DBMS.

The following table shows the acceptable Open Server datatypes that the DB2 access service can convert into corresponding DB2 datatypes. To save space in this table, the CS_ PREFIX is removed from Open Server datatypes, and similar DB2 datatypes are combined.

Table 9-1: Open Server to DB2 datatype conversions

From Open Server datatypes (below)	**To DB2 datatypes (across)**				
	CHAR *VARCHAR*	*INT* *SMALLINT*	*DECIMAL* *FLOAT* *REAL*	*DATE* *TIME* *TIMESTAMP*	*CHAR FOR BIT DATA* *VARCHAR FOR BIT DATA*
CHAR	X	X	X	X	X
VARCHAR		X	X	X	X
TEXT	X	X	X	X	X
BINARY	X				X
VARBINARY	X				X
IMAGE	X				X
BIT	X	X	X		
SMALLINT	X	X	X		
INTEGER	X	X	X		
DECIMAL	X	X	X		
MONEY	X	X	X		
MONEY4	X	X	X		
REAL	X	X	X		
FLOAT	X	X	X		
DATETIME	X			X	
DATETIME4	X			X	

Datatype conversion for *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	
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.

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 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 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.

Source input datatypes	Target (destination) datatypes	Error conditions for bulk copy
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).
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.

Source input datatypes	Target (destination) datatypes	Error conditions for bulk copy
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.
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.

Source input datatypes	Target (destination) datatypes	Error conditions for bulk copy
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 statement and 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)
- Replace existing rows (replace or truncate)
- 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 any other arbitrary SQL statement

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 bulk 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 168.

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 SQL_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

Open Server datatype	Default	Effects (by qualifier)			
		?C	**?N**	**?D**	**?T**
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	Quote	Quote	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’*	*‘yy/mm/dd’*	*‘hh:mm:ss’*	n/a

Open Server datatype	Default	Effects (by qualifier)			
		?y	?d	?t	?x
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 2 digits of year.
yyyy	All 4 digits of year.
mm	Month or minute (recognized by context).
mon	Month abbreviation in 3 characters: 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

To obtain information about database objects, you need to access the database catalog. Catalog stored procedures (CSPs) provide this catalog access. This chapter describes how to use CSPs to access the DB2 catalog.

This chapter contains the following topics:

General information

This section provides an overview of CSPs.

Why use CSPs

The catalog structures for DB2 and Adaptive Server are different. If you have client applications written to access the Adaptive Server catalog, you may need to re-code the client application queries to send those queries directly to the DB2 system tables. To avoid modifying your database-specific applications, you can use CSPs to access catalog information. CSPs are compatible with the catalog interface for the Open Database Connectivity (ODBC) Application Program Interface (API).

CSP properties

Configuration properties specify the objects about which an access service returns information from CSPs. For information about configuring these properties, see Chapter 2, "Creating and Configuring DB2 Access Services."

CSP set statements

You can use set statements to change operating values for a client connection. For information about CSP set statements, see "Catalog Stored Procedure properties" on page 71.

How the DB2 access service assembles stored procedure information

When you invoke a CSP, the DB2 access service creates a SQL statement that returns a result set that matches, as closely as possible, the Microsoft ODBC 2.0 specification results. For detailed information about the Microsoft ODBC 2.0 specification, see the *Microsoft ODBC 2.0 Programmer's Reference and SDK Guide*.

Because the DB2 system catalog is significantly different from the ODBC specification, an exact match is impossible. To make the results match, the DB2 access service performs additional computations as it returns the rows. In some cases, the SQL statement is quite complex and must assemble the requested information from multiple sources.

In a few cases, such as sp_stored_procedures, sp_sproc_columns, and sp_server_info, information is stored in tables that were created during installation of DirectConnect.

Some information is static, depending upon the version of DB2 you are using. In these cases, the DB2 access service uses memory tables to improve the speed of operation. The following CSPs and system procedures use memory tables:

- sp_capabilities
- sp_datatype_info
- sp_helpserver
- sp_sqlgetinfo

Supported CSPs

The following table shows CSPs that a DB2 access service supports.

Table 11-1: Supported CSPs

CSP	Information retrieved by the CSP
sp_capabilities	Returns the SQL capabilities of a DB2 access service
sp_column_privileges	Column privilege information for one table
sp_columns	Column descriptions for a table
sp_databases	List of available databases
sp_datatype_info	Datatype descriptions
sp_fkeys	Foreign and primary key relationships
sp_pkeys	Primary key information for a single table
sp_server_info	Server terms, limits, and capabilities
sp_special_columns	Additional column information
sp_sproc_columns	Attributes of procedures input and return parameters
sp_statistics	Statistics and indexes for one table
sp_stored_procedures	List of available procedures
sp_table_privileges	Table privilege information for one table
sp_tables	List of aliases, synonyms, tables, views, and system tables

Coding instructions

This section includes general coding information that applies to all CSPs and system procedures.

Parameters

CSPs and system procedures have both optional and required parameters. Required parameters must have values supplied; optional parameters default to predefined values.

The following rules apply to CSP and system procedure parameters:

- Both positional and named parameters are supported, but not in the same statement.
- Parameter values can be enclosed in double quotes. Parameter values enclosed in quotes must be in the correct case for the target.

- Object names (table names, column names, and index names) can be created using lowercase letters. The target database automatically converts object names to uppercase unless the object names are enclosed in double quotes.

Syntax

A client application can initiate a CSP or system procedure by issuing any of the following statements:

```
sp_name parm1, parm2, . . .
exec sp_name parm1, parm2, . . .
execute sp_name parm1, parm2, . . .
```

where:

- *sp_name* is the name of the stored procedure (for example, sp_columns).
- *parm1* and *parm2* are parameter values required or desired for that stored procedure.

Coding examples

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

You can specify the parameters for a CSP or system procedure in one of the following forms:

- Supply all of the parameters:

  ```
  sp_columns publishers, “dbo”, “pubs2”, “pub_id”
  ```

- Use “null” or a comma as a placeholder:

  ```
  sp_columns publishers, null, null,”pub_id”
  sp_columns publishers, , , “pub_id”
  ```

- Supply one or more parameters in the following form:

  ```
  @parameter_name = value
  ```

  For example, to find information about a particular column, issue the following statement:

  ```
  sp_columns @table_name = publishers, @column_name =
  "pub_id"
  ```

The parameter names in the syntax statement must match the parameter names defined by the CSP.

You cannot use this named parameter form if you process a CSP as an RPC event.

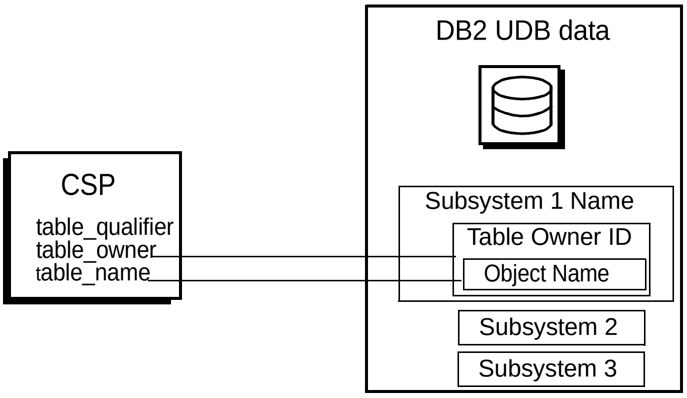
Table name, owner, and qualifier parameters

This section explains how the parameters *table_name*, *table_owner*, and *table_qualifier* are used in this product.

- *table_name* is the name of the database object about which you want to retrieve catalog information.
- *table_owner* is the owner of the database object about which you want to retrieve catalog information. If you do not specify a value, the DB2 access service returns information that matches the supplied criteria for all authorization IDs.
- *table_qualifier* is ignored. Leave blank or set to NULL.

The following figure shows how CSP parameters relate to the DB2 subsystem.

Figure 11-1: CSP parameters and DB2



Wildcard-character search patterns

The percent (%) wildcard can be used in parameters that allow wildcard-character search patterns. This wildcard represents any string of zero or more characters.

If the percent (%) character is used in parameters that do not allow wildcard-character search patterns, the DB2 access service rejects the character and issues an error message.

The following table shows some examples of the percent (%) wildcard character and its use:

Table 11-2: Wildcard examples

Sample string	Matches
%A%	All names that contain the letter "A"; for example, A, AT, CAT
%	All names

Escape character

To use a wildcard character as a literal, precede it with an @ (at) sign. If the parameter normally accepts the wildcard character, you can mix the percent (%) wildcard character with escaped wildcard characters (@%) interpreted as literals. If the parameter does not accept the wildcard character, an @ (at) sign (@) must precede the wildcard character to use the character as a literal.

sp_column_privileges

Description

Returns column privilege information for a single database object.

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. Aliases are not supported. Views are supported but do not include alter or index privileges.

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 function corresponds to the ODBC function SQLColumnPrivileges.
- Information is based on the SYSCOLAUTH, SYSCOLUMNS, and SYSTABAUTH system catalog tables.

Results

sp_column_privileges returns one row for each privilege a user has on a column in a table. Results are ordered by the following columns:

- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- PRIVILEGE

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_column_privileges

Column name	Datatype	Description
TABLE_QUALIFIER	varchar (128)	Always NULL
TABLE_OWNER	varchar (128)	Authorization ID
TABLE_NAME	varchar (128) NOT NULL	Name of the object about which privilege information is returned
COLUMN_NAME	varchar (128) NOT NULL	Column name
GRANTOR	varchar (128)	Identifies the user who granted this privilege
GRANTEE	varchar (128) NOT NULL	Identifies the user to whom this privilege was granted
PRIVILEGE	varchar (128) NOT NULL	Identifies the privilege granted to the grantee on this column as one of the following values: • SELECT if the grantee is authorized to select rows in the associated object • UPDATE if the grantee is authorized to insert and update rows in the associated object
IS_GRANTABLE	varchar (3)	Indicates whether the grantee is authorized to grant privilege on this column to other users; always NULL

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. Use the wildcard character to request information about more than one table. Aliases are not supported.

table_owner
is the owner of the database object about which column information is requested. Use the wildcard character to request information about tables owned by more than one user. If you do not specify a table owner, sp_columns looks first for tables owned by the current user and then for tables owned by the database owner.

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. Leave empty or set to NULL to request information about all columns in the table or tables.

Usage

- If *column_name* is provided, sp_columns returns information only for the column or columns that match.
- This function corresponds to the ODBC function SQLColumns.
- Information is based on the SYSCOLUMNS and SYSSYNONYMS system catalog tables.

Results

sp_columns returns one row containing a description of each column in a table. Results are ordered by the following columns:

- TABLE_OWNER
- TABLE_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 11-4: Result set for sp_columns

Column	Datatype	Description
TABLE_QUALIFIER	varchar(128)	Always NULL
TABLE_OWNER	varchar(128)	Table owner identifier
TABLE_NAME	varchar(128) NOT NULL	Table name
COLUMN_NAME	varchar(128) NOT NULL	Column name
DATA_TYPE	smallint NOT NULL	Integer code for the ODBC datatype
TYPE_NAME	varchar(128) NOT NULL	String representing the datatype name in the target database
PRECISION	int	Number of significant digits of the column on the target database
LENGTH	int	Length of the column in bytes

Column	Datatype	Description
SCALE	smallint	Number of digits to the right of the decimal point
RADIX	smallint	Base for numeric types
NULLABLE	smallint NOT NULL	Indicates whether the column accepts NULL values: • 0 SQL_NO_NULLS if the column does not accept NULL values • 1 SQL_NULLABLE if the column accepts NULL values • 2 SQL_NULLABLE_UNKNOWN if it is not known if the column accepts NULL values
REMARKS	varchar(254)	A description of the column
SS_DATA_TYPE	smallint	The Adaptive Server datatype name
COLID	smallint	The column ID number
REMOTE_DATA_TYPE	int	An integer representing the underlying target database datatype (composite value)

ODBC Datatypes

The following table describes the DB2 datatypes and matching ODBC integer identifiers that are returned in the TYPE_NAME and DATA_TYPE columns of the sp_columns, sp_datatype_info, sp_special_columns, and sp_sproc_columns result sets.

Table 11-5: ODBC datatypes

DB2 datatype (TYPE_NAME)	Target datatype maximum physical length	ODBC type	ODBC integer ID (DATA_ TYPE)	DB2 datatype description
CHARACTER() FOR BIT DATA	254	SQL_BINARY	-2	Fixed length character for bit data
VARCHAR() FOR BIT DATA	254	SQL_VARBINARY	-3	Variable length character for bit data
LONG VARCHAR FOR BIT DATA	32714	SQL_LONGVARBINARY	-4	Variable length character for bit data
CHARACTER()	254	SQL_CHAR	1	Fixed length character
VARCHAR()	254	SQL_VARCHAR	12	Variable length character
LONG VARCHAR()	32714	SQL_LONGVARCHAR	-1	Variable length character

DB2 datatype (TYPE_NAME)	Target datatype maximum physical length	ODBC type	ODBC integer ID (DATA_ TYPE)	DB2 datatype description
CHARACTER() FOR MIXED DATA	254	SQL_BINARY	-2	Fixed length character (DBCS or SBCS)
VARCHAR() FOR MIXED DATA	254	SQL_VARBINARY	-3	Variable length character (DBCS or SBCS)
LONG VARCHAR() FOR MIXED DATA	32714	SQL_LONGVARBINARY	-4	Variable length character (DBCS or SBCS)
GRAPHIC()	127	SQL_BINARY	-2	Fixed length graphic (DBCS)
VARGRAPHIC()	127	SQL_VARBINARY	-3	Variable length graphic (DBCS)
LONG VARGRAPHIC	16357	SQL_LONGVARBINARY	-4	Variable length graphic (DBCS)
SMALLINT	2	SQL_SMALLINT	5	2-byte binary integer
INTEGER	4	SQL_INTEGER	4	4-byte binary integer
REAL	4	SQL_REAL	7	4-byte floating point
FLOAT()	4	SQL_REAL	7	4-byte floating point with a precision less than 22
FLOAT()	8	SQL_DOUBLE	8	8-byte floating point with a precision equal to or greater than 22
DOUBLE PRECISION	8	SQL_DOUBLE	8	8-byte floating point
DECIMAL()	31	SQL_DECIMAL	3	Packed decimal number
NUMERIC	31	SQL_NUMERIC	2	Zoned decimal number
DATE	10	SQL_DATE	9	Date
TIME	8	SQL_TIME	10	Time
TIMESTAMP	26	SQL_DATETIME	11	Timestamp

REMOTE_DATATYPE

The REMOTE_DATATYPE column contains a 32-bit composite datatype value that represents the target database datatype.

The following table describes the datatype value.

Table 11-6: REMOTE_DATATYPE value

Bit(s)	Description
Bits 0-7	ODBC (target) datatype (can be extended for types not defined in ODBC)
Bit 8	Returns 1 if nullable, 0 if not nullable
Bit 9	Returns 1 if case sensitive, 0 if not case sensitive
Bits 10, 11	Always returns 10 (binary) meaning updatability unknown
Bits 12, 13	Reserved, always returns 00 (binary)
Bits 14, 15	Returns the following: • 01 (binary) meaning NEWODBCDATATYPE (used for all except REAL) • 10 (binary) meaning NEWUSERTYPE (used for REAL)
For numeric types: Bits 16–23	Precision
Bits 24–31	Scale
For 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
Information is based on the SYSDATABASE system catalog table.

Results

sp_databases returns a list of databases available to the client. Results are ordered by DATABASE_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 11-7: Result set for sp_databases

Column	Datatype	Description
DATABASE_NAME	varchar(32) NOT NULL	Name of an available database
DATABASE_SIZE	int	Size of the named database in kilobytes, NULL
REMARKS	varchar(254)	Always NULL

sp_datatype_info

Description

Returns information about a particular datatype or about all supported datatypes.

Syntax

sp_datatype_info [*data_type*]

Parameters

data_type
is the ODBC code number for the specified datatype about which sp_datatype_info returns information. See Table 11-5 on page 185 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 function corresponds to the ODBC function SQLGetTypeInfo.

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 DatatypeInfo property specifies whether information is returned about T-SQL datatypes or target database datatypes. For configuration information, see “DatatypeInfo” on page 28.

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 11-8: 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.

Column	Datatype	Description
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.
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 using 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. Views, aliases, and wildcard-character search patterns are not supported. You must specify either this parameter or the fktable_name parameter, or both. Views and aliases are not supported.

pktable_owner
is the owner of the table containing the primary key. Views, aliases, and wildcard-character search patterns are not supported. If you do not specify this parameter, sp_fkeys looks first for a table owned by the current user and then for a table owned by the database owner.

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. Views and aliases are not supported.

fktable_owner
is the owner of the table containing the foreign key. Wildcard-character search patterns are not supported. If you do not specify this parameter, sp_fkeys looks first for a table owned by the current user and then for a table owned by the database owner.

fktable_qualifier
is ignored. Leave blank or set to NULL.

Usage

- This function corresponds to the ODBC function SQLForeignKeys.
- Information is based on the SYSCOLUMNS, SYSFOREIGNKEYS, SYSINDEXES, SYSRELS, and SYSSYNONYMS system catalog tables.
- For information about creating a foreign key, see the appropriate *IBM DATABASE 2 SQL Reference* manual.

Results

sp_fkeys returns a row for each column that is part of the foreign key or primary key in a primary key/foreign key relationship.

Results are ordered by the following columns:

- PKTABLE_OWNER
- PKTABLE_NAME
- KEY_SEQ

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 11-9: Result set for sp_fkeys

Column	Datatype	Description
PKTABLE_QUALIFIER	varchar(128)	NULL.
PKTABLE_OWNER	varchar(128)	Primary key table owner.
PKTABLE_NAME	varchar(128) NOT NULL	Primary key table name.
PKCOLUMN_NAME	varchar(128) NOT NULL	Primary key column name.
FKTABLE_QUALIFIER	varchar(128)	NULL.
FKTABLE_OWNER	varchar(128)	Foreign key table owner.
FKTABLE_NAME	varchar(128) NOT NULL	Foreign key table name.
FKCOLUMN_NAME	varchar(128) NOT NULL	Foreign key column name.
KEY_SEQ	smallint NOT NULL	Column sequence number in key (starting with 1).
UPDATE_RULE	smallint	Action to be applied to the foreign key when the SQL operation is update: • 0 means cascade. • 1 means restrict. • 2 means set null. • NULL means not applicable to the target database.
DELETE_RULE	smallint	Action to be applied to the foreign key when the SQL operation is delete: • 0 means cascade. • 1 means restrict. • 2 means set null. • NULL means not applicable to the target database.
FK_NAME	varchar(128)	Foreign key identifier; NULL if not applicable to the target database.
PK_NAME	varchar(128)	Primary key identifier; NULL if not applicable to the target database.

sp_pkeys

Description

Returns primary key information for the specified table or tables.

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. Views and aliases are not supported.

table_owner
is the owner of the table. Wildcard-character search patterns are not supported. If you do not specify this parameter, sp_fkeys looks first for a table owned by the current user and then for a table owned by the database owner.

table_qualifier
is ignored. Leave blank or set to NULL.

Usage

- This function corresponds to the ODBC function SQLPrimaryKeys.
- Information is based on the SYSINDEXES, SYSKEYS, and SYSSYNONYMS system catalog tables.
- For information about creating a foreign key, see the appropriate *IBM DATABASE 2 SQL Reference*.

Results

sp_pkeys returns a row for each column in the primary key. Results are ordered by:

- TABLE_OWNER
- TABLE_NAME
- KEY_SEQ

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 11-10: Result set for sp_pkeys

Column	Datatype	Description
TABLE_QUALIFIER	varchar(128)	NULL
TABLE_OWNER	varchar(128)	Primary key table owner (authorization ID)
TABLE_NAME	varchar(128) NOT NULL	Primary key table name
COLUMN_NAME	varchar(128) NOT NULL	Primary key column name
KEY_SEQ	smallint NOT NULL	Sequence number of the column in a multi-column primary key
PK_NAME	varchar(128)	Primary key identifier; NULL if not applicable to the target database

sp_server_info

Description

Returns a list of attribute names and matching values for the target DBMS.

Syntax

sp_server_info [*attribute_id*]

Parameters

attribute_id
is the integer ID of the attribute. Wildcard-character search patterns are not supported.

Usage

- If the *attribute_id* parameter is not provided, sp_server_info returns information about all attributes.
- This function does not correspond to any ODBC function, but returns some of the information returned by SQLGetInfo.

Results

sp_server_info returns a list of the requested attributes and their values.

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 11-11: Result set for sp_server_info

Column	Datatype	Description
ATTRIBUTE_ID	int NOT NULL	Numeric identifier of the attribute
ATTRIBUTE_NAME	varchar(60)	Attribute name
ATTRIBUTE_VALUE	varchar(254)	Attribute value

sp_special_columns

Description

Retrieves the following information about columns within a specified table or view:

- The optimal set of columns that uniquely identify a row in the table or view
- A list of the columns that are automatically updated when any value in the row is updated

Syntax

sp_special_columns *table_name* [, *table_owner*]
[, *table_qualifier*] [, *col_type*]

Parameters

table_name
is the name of the table. Views, aliases, and wildcard-character search patterns are not supported. Views and aliases are not supported.

table_owner
is the owner of the table. Views, aliases, and wildcard-character search patterns are not supported. If you do not specify this parameter, sp_special_columns looks first for a table owned by the current user and then for a table owned by the database owner.

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 function corresponds to the ODBC function SQLSpecialColumns.

- Information is based on the SYSINDEXES, SYSKEYS, and SYSCOLUMNS system catalog tables.

Results

sp_special_columns returns information about the columns that uniquely identify a row in a table.

The result set consists of a row for each column of an index that uniquely identifies each row of the table. If there are multiple unique indexes on a table, the one that is described by the result set is the first that exists in the following list:

- A primary key with clustered index
- A primary key without clustered index
- A unique, clustered index
- A unique, non-clustered index

The result set is ordered by the column name in the index.

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 11-12: Result set for sp_special_columns

Column	Datatype	Description
SCOPE	smallint NOT NULL	Actual scope of the row ID: • 0 SQL_SCOPE_CURROW • 1 SQL_SCOPE_TRANSACTION
COLUMN_NAME	varchar(128) NOT NULL	Column name
DATA_TYPE	smallint NOT NULL	ODBC datatype to which all columns of this type are mapped
TYPE_NAME	varchar(128) NOT NULL	Name of the target database datatype that corresponds to the ODBC datatype in the DATA_TYPE column
PRECISION	int	Maximum precision for the datatype in the target database; NULL if precision is not applicable
LENGTH	int	Length of the column in bytes
SCALE	smallint	Number of digits to the right of the decimal point; NULL if scale is not applicable
PSEUDO_COLUMN	smallint	Indicates whether the column is a pseudo-column; the DB2 access service always returns: 0 SQL_PC_UNKNOWN

sp_sproc_columns

Description

Returns descriptive information for the input and return parameters for stored procedures in the current environment.

Syntax

sp_sproc_columns *sp_name* [, *sp_owner*]
 [, *sp_qualifier*] [, *column_name*]

Parameters

sp_name
is the name of the stored procedure. 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 stored procedures owned by more than one user. If you do not specify this parameter, sp_sproc_columns looks first for a procedure owned by the current user and then for a procedure owned by the database owner.

sp_qualifier
is ignored. Leave blank or set to NULL.

column_name
is the set of columns to be included in the result set. Use the wildcard character to request information about more than one column. If you do not supply a *column_name* parameter, sp_sproc_columns returns information about all columns for the stored procedure.

Usage

- The access service selects information from the SYSPROCCOLUMNS table. The *cspdb2.sql* script creates this table during installation of DirectConnect. However, you need to update the SYSPROCCOLUMNS table manually.
- This function corresponds to the ODBC function SQLProcedureColumns.

Results

sp_sproc_columns returns a list of available procedures. Results are ordered by the following columns:

- PROCEDURE_OWNER
- PROCEDURE_NAME
- COLUMN_TYPE

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 for sp_sproc_columns.

Table 11-13: Result set for sp_sproc_columns

Column	Datatype	Description
PROCEDURE_QUALIFIER	varchar(128)	Always NULL
PROCEDURE_OWNER	varchar(128)	Value from the corresponding column of SYSPROCCOLUMNS table
PROCEDURE_NAME	varchar(128) NOT NULL	Name of the stored procedure
COLUMN_NAME	varchar(128) NOT NULL	Name of the input parameter or result set column

Column	Datatype	Description
COLUMN_TYPE	smallint NOT NULL	Type of data in this procedure column: • 1 SQL_PARAM_INPUT – the procedure column is an input parameter • 3 SQL_RESULT_COL – the procedure column is a result set column
DATA_TYPE	smallint NOT NULL	Integer code for the ODBC SQL datatype equivalent of the target database datatype for this procedure column
TYPE_NAME	varchar(128) NOT NULL	String representing the datatype name in the target database
PRECISION	int	Precision of the procedure column on the target database; NULL if precision is not applicable
LENGTH	int	Length of the column in bytes
SCALE	smallint	Number of digits to the right of the decimal point; NULL if scale is not applicable
RADIX	smallint	Base for numeric types; NULL if radix is not applicable
NULLABLE	smallint	Indicates whether the procedure column accepts NULL values: • 0 – the column does not accept NULL • 1 – the column accepts NULL • 2 – it is not known if the column accepts NULL values
REMARKS	varchar(254)	Description of the procedure column

sp_statistics

Description

Returns statistics information for a single table and the indexes associated with that table.

Syntax

sp_statistics *table_name* [, *table_owner*]
 [, *table_qualifier*] [, *index_name*] [, *is_unique*]

Parameters

table_name
is name of the table. Views, aliases, and wildcard-character search patterns are not supported. Views and aliases are not supported.

table_owner
is the owner of the database object about which column privilege information is requested. Wildcard-character search patterns are not supported. If you do not specify this parameter, sp_statistics looks first for a table owned by the current user and then for a table owned by the database owner.

table_qualifier
is ignored. Leave blank or set to NULL.

index_name
is the name of the index. Wildcard-character search patterns are not supported.

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

Usage

- If *index_name* is specified, sp_statistics returns only information about that index.
- This function corresponds to the ODBC function SQLStatistics.

Results

sp_statistics returns information about the named table. Results are ordered by the following columns:

- NON_UNIQUE
- TYPE
- INDEX_QUALIFIER
- INDEX_NAME
- SEQ_IN_INDEX

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 11-14: Result set for sp_statistics

Column	Datatype	Description
TABLE_QUALIFIER	varchar(128)	Always NULL.
TABLE_OWNER	varchar(128)	Table owner authorization ID.
TABLE_NAME	varchar(128) NOT NULL	Name of the table or view.

Column	Datatype	Description
NON_UNIQUE	smallint	Indicates whether the index permits duplicate values: • 0 (FALSE) means the index prohibits duplicate values. • 1 (TRUE) means the index allows duplicate values. • NULL is returned if TYPE is SQL_TABLE_STAT.
INDEX_QUALIFIER	varchar(128)	Always NULL
INDEX_NAME	varchar(128)	Index name; NULL is returned if TYPE is SQL_TABLE_STAT.
TYPE	smallint NOT NULL	Type of information returned: • 0 SQL_TABLE_STAT means statistics for a table. • 1 SQL_INDEX_CLUSTERED means a clustered index. • 2 SQL_INDEX_HASHED means a hashed index. • 3 SQL_INDEX_OTHER means another type of index.
SEQ_IN_INDEX	smallint	Sequence of the column in the index (the first column is 1); NULL is returned if TYPE is SQL_TABLE_STAT.
COLUMN_NAME	varchar(128)	Column name; NULL is returned if TYPE is SQL_TABLE_STAT.
COLLATION	char(1)	Sort sequence for the column: • A means ascending. • D means descending. • NULL is returned if TYPE is SQL_TABLE_STAT.
CARDINALITY	int	Cardinality of the table or index: • Number of rows in the table if TYPE is SQL_TABLE_STAT. • Number of unique values in the index if TYPE is not SQL_TABLE_STAT. • NULL if the value is not available from the target database.
PAGES	int	Number of pages used to store the index or table: • Number of pages used to store the table if TYPE is SQL_TABLE_STAT. • Number of pages used to store the index if TYPE is not SQL_TABLE_STAT. • NULL if this information is not available from the target database.
FILTER_CONDITION	varchar(128)	If the index is a filtered index, this is the filter condition; if the filter condition cannot be determined, this is an empty string. NULL is returned if the index is not a filtered index or TYPE is SQL_TABLE_STAT.

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. If left blank, sp_stored_procedures returns information for all procedures.

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 ignored. Leave blank or set to NULL.

Usage

- The DB2 access service selects information from the SYSPROCEDURES table. The *cspdb2.sql* script creates this table during installation of DirectConnect.
- This function corresponds to the ODBC function SQLProcedures.

Results

sp_stored_procedures lists and describes stored procedures. Results are ordered by the following columns:

- *PROCEDURE_QUALIFIER*
- *PROCEDURE_OWNER*
- *PROCEDURE_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 describes the result set.

Table 11-15: Result set for sp_stored_procedures

Column	Datatype	Description
PROCEDURE_QUALIFIER	varchar(128)	Always NULL.
PROCEDURE_OWNER	varchar(128)	Procedure owner.
PROCEDURE_NAME	varchar(128) NOT NULL	Procedure name.
NUM_INPUT_PARAMS	int NOT NULL	Number of input parameters in the stored procedure. -1 means the number of input parameters is unknown.

Column	Datatype	Description
NUM_OUTPUT_PARAMS	int NOT NULL	Number of return parameters in the stored procedure. -1 means the number of return parameters is unknown.
NUM_RESULT_SETS	int NOT NULL	Number of result sets returned by the stored procedure. -1 means the number of result sets is unknown.
REMARKS	varchar(254)	A description of the procedure.
PROCEDURE_TYPE	smallint	Defines the procedure type: • 0 SQL_PT_UNKNOWN if it cannot be determined whether the procedure returns a value. • 1 SQL_PT_PROCEDURE if the returned object is a procedure; it does not have a return value. • 2 SQL_PT_FUNCTION if the returned object is a function; it has a return value.

sp_table_privileges

Description

Returns privilege information for one or more database objects.

Syntax

sp_table_privileges *table_name* [, *table_owner*]
 [, *table_qualifier*]

Parameters

table_name
is the name of the table. Use the wildcard character to request information about more than one table. Aliases are not supported.

table_owner
is the owner of the database object about which column privilege information is requested. Use the wildcard character to request information about tables owned by more than one user. If you do not specify this parameter, sp_table_privileges looks first for a table owned by the current user and then for a table owned by the database owner.

table_qualifier
is ignored. Leave blank or set to NULL.

Usage

- The DB2 access service selects information from the SYSTABAUTH system catalog table.
- This function corresponds to the ODBC function SQLTablePrivileges.

Results

sp_table privileges returns a list of one or more database objects with privilege information about each. Results are ordered by the following columns:

- TABLE_OWNER
- TABLE_NAME
- PRIVILEGE

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 11-16: Result set for sp_table_privileges

Column name	Datatype	Notes
TABLE_QUALIFIER	varchar (128)	Always NULL.
TABLE_OWNER	varchar (128)	Table owner identifier (authorization ID).
TABLE_NAME	varchar (128) NOT NULL	Name of the database object about which privilege information is returned.
GRANTOR	varchar (128)	Identifies the user who granted this privilege; NULL if not applicable to the target database.
GRANTEE	varchar (128) NOT NULL	Identifies the user to whom this privilege was granted.
PRIVILEGE	varchar (128) NOT NULL	Identifies the privilege granted to the grantee on this object as one of the following values: • SELECT if the grantee is authorized to select rows in the associated object. • INSERT if the grantee is authorized to insert rows into the associated object. • UPDATE if the grantee is authorized to update rows in the associated object. • REFERENCES if the grantee is authorized to refer to one or more columns of the table within a constraint (for example: unique, referential, or table check constraint).
IS_GRANTABLE	varchar (3)	Indicates whether the grantee is authorized to grant privilege on this object to others users with one of the following values: • YES if the grantee can grant this privilege to others. • NO if the grantee cannot grant this privilege to others. • NULL if it is unknown or not applicable to the target database.

sp_tables

Description

Returns a list of objects stored in the database.

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
is ignored. Leave empty or set to NULL.

table_type
is a list of values, separated by commas, requesting information about all objects of a specific type(s) as follows:

"'TABLE', 'SYSTEM TABLE', 'VIEW', 'ALIAS', 'SYNONYM'"

Note You must enclose each table type with single quotation marks, and enclose the entire parameter with double quotation marks. Enter table types in uppercase.

Usage

- The DB2 access service CSP configuration properties limit the set of object types for which information is returned. Using the *table_type* parameter, the object types can be more limited, but not less limited. For example, if the DB2 access service CSP configuration properties indicate that only tables and views are included and the *table_type* parameter specifies all object types are included, sp_tables only returns information about tables and views. For information, see "Catalog Stored Procedure properties" on page 24.
- The DB2 access service selects information from the SYSSYNONYMS, SYSTABAUTH, and SYSTABLES system catalog tables.
- This function corresponds to the ODBC function SQLTables.

Results

sp_tables returns a list of database objects. Results are ordered by the following columns:

- *TABLE_TYPE*
- *TABLE_OWNER*
- *TABLE_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 11-17: Result set for sp_tables

Column	Datatype	Description
TABLE_QUALIFIER	varchar(128)	Always NULL
TABLE_OWNER	varchar(128)	Table owner
TABLE_NAME	varchar(128)	Name of the object about which information is returned
TABLE_TYPE	varchar(128) NOT NULL	One of the following: • ‘ALIAS’ • ‘SYNONYM’ • ‘SYSTEM TABLE’ • ‘TABLE’ • ‘VIEW’
REMARKS	varchar(254)	A description of the table or NULL

CHAPTER 12

Retrieving Information with System Procedures

System procedures are Sybase-supplied stored procedures that return information about the DB2 access service and the target database. This chapter describes the system procedures that a DB2 access service supports.

If a DB2 access service cannot support a procedure, that procedure returns a correctly formatted result set containing zero rows.

A client application initiates a system procedure in the same way that it initiates a CSP. For instructions to code both CSP and system procedures, see “Coding instructions” on page 178.

This chapter contains the following topics:

Topic	Page
sp_capabilities	207
sp_groups	211
sp_helpserver	211
sp_password	212
sp_sqlgetinfo	214

sp_capabilities

Description

Returns the SQL capabilities of a DB2 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 a DB2 access service during normal query processing.

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 the ID and values for several DB2 access service functional capabilities:

Table 12-2: sp_capabilities information

ID	Capability	Value description
101	SQL syntax	1=Sybase T-SQL supported 2=DB2 SQL supported
102	Join handling	0=Unsupported 1=No outer join supported 2=T-SQL support 3=Oracle supported
103	Aggregate handling	0=Unsupported 1=ANSI 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-style supported 2=T-SQL supported
107	Bulk insert handling	0=Unsupported 1=Supported
108	Text and image handling	0=Unsupported 1=Text, no textptr 2=Text and textptr
109	Transaction handling	0=Unsupported 1=Local supported 2=Two-phase commit supported
110	Text pattern handling	0=Unsupported 1=Pattern (text) supported
111	order by	0=Unsupported 1=Supported
112	group by	0=Unsupported 1=ANSI supported 2=T-SQL 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 Non-zero=Escape_char(s)

ID	Capability	Value description
117	Union handling	0=Unsupported 1=Supported
118	String functions	0=Unsupported 1=Substring supported 2=Oracle subset supported 3=T-SQL supported
119	Expression handling	0=Unsupported 1=ANSI supported 2=T-SQL supported
120	Character truncation	0=Fixed length character parameters may contain trailing blanks 1=Fixed length character parameters will not contain trailing blanks
121	Language events	0=Unsupported 1=T-SQL DML without datetime in the where clause supported 2=T-SQL DML supported
122	Date functions	0=Unsupported 1=T-SQL date functions supported
123	Math functions	0=Unsupported 1=Oracle functions supported 2=T-SQL math functions supported
124	T-SQL convert functions	0=Unsupported 1=Supported
125	T-SQL delete/update	0=Sybase extensions not supported 1=Sybase extensions supported
126	Insert/select handling	0=Unsupported 1=Supported
127	Subquery handling	0=Unsupported 1=Supported
128	IN/NOT IN support	0=Unsupported 1=Supported
129	CASE support	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

ID	Capability	Value description
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_groups

Description
Returns a list of groups to which the current user belongs.

Syntax
sp_groups

Parameters
None.
This procedure does not allow parameters.

Usage
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
Provides the user with a report containing the following rows of information:

- Row 1: the version of Open Server currently in use
- Row 2: the version of DirectConnect server currently in use
- Row 3: the version of the DB2 Access Service Library currently in use
- Row 4: the version of the DBMS with which an access service is associated

Syntax
sp_helpserver

Parameters
This procedure does not allow parameters.

Usage

Results

The following table shows the sp_helpserver result set.

Table 12-4: Result set for sp_helpserve

Column name	Datatype	Description
SERVER_PROPERTY	varchar(32)	Server property name
PROPERTY_VALUE	varchar(32)	Server property value

sp_password

Description

Changes or queries the current status of the user’s password.

Syntax

To change the user’s password, use the following syntax:
sp_password *old_password*, *new_password*, [*login_name*]

To query the current status of the user’s password, use the following syntax:
sp_password

Parameters

old_password
is the current password for the user of this procedure or login name.

new_password
is the new password for the user. The new password must conform to the rules required on the DBMS in which the password change is to take effect.

login_name
is the login name of the user whose password is to be changed.

Note The mainframe security system rejects the request if the user lacks the authority to change another user’s password.

Usage

- You can program your client applications to use sp_password to avoid password expiration problems. Program an application to issue sp_password upon application start-up, and prompt the user for a new password if the existing password is close to expiration. Also, you can build a simple application that reads user IDs and passwords from a file and executes sp_password to make the appropriate modifications.

- For DB2, the DB2 access service invokes the CICS transaction called Password Expiration Manager (PEM). PEM is a password management program that IBM provides with CICS 3.3 through an optional PTF UN90057. PEM is available only for connections to the mainframe using LU 6.2. It is not available for TCP/IP connections.

Note To change the setting of the CICS SIT table property, ask the CICS system programmer and the external security manager. The SIT property defines the intersystem refresh delay, which determines how long users remain signed on to the host when running transactions with the InterSystem Communication (ISC) setting. Its setting can affect the ability of users to log in more than once or to run multiple host transactions from TRS within the defined time period. By default, the delay is set to 30 minutes. Sybase recommends setting ISRDELAY=0.

Results

The following table shows the result set:

Table 12-5: Result set for sp_password

Column	Datatype	Possible return values and descriptions
RETURN_CODE	smallint	0 = The PEM operation returned no errors. 1 = The user ID was not found on the host. 2 = The password is incorrect. 3 = No new password was specified and the old password expired. 4 = The host security system rejected the new password. 5 = A security function failure occurred. Your password account may be revoked. 6 = An invalid request was made to PEM. The new or old password may be in an unacceptable format. 7 = General security error. 8 = Password change completed but signon failed.
CURRENT_DATE	datetime	Date and time of current successful signon.
LAST_DATE	datetime	Date and time of last successful signon.
EXPIRE_DATE	datetime	Date and time password will expire.
REVOKE_COUNT	smallint	Number of unsuccessful signon attempts since the last successful signon with this user ID.

sp_sqlgetinfo

Description

Provides information about SQL parameters that the target database supports for the specified attribute.

Syntax

sp_sqlgetinfo [*attribute_name*]

Parameters

attribute_name
is the name of the attribute about which information is requested. Wildcard characters are supported.

Usage

- If the attribute is not found in the internal table, the DB2 access service returns an empty result set.
- If a parameter is not provided, the DB2 access service returns a result set of all supported attributes.
- This stored procedure provides SQL grammar, syntax, and capabilities that are supported on the target database.

- The DB2 access service stores this information in a file that you can modify if necessary. The SQLInformationFile property specifies the path and file name. For more information, see “SQLInformationFile” on page 28.

Results

The following table shows the sp_sqlgetinfo result set.

Table 12-6: Result set for sp_sqlgetinfo

Column Name	Datatype	Description
ATTRIBUTE_ID	int not NULL	The numeric identifier for the attribute
ATTRIBUTE_NAME	varchar(30) not NULL	The name of the attribute
ATTRIBUTE_VALUE	varchar(255) NULL	The value of the attribute

CHAPTER 13

RSPs and Host-Resident Requests

This chapter describes how to create, execute, and delete an RSP and a host-resident request. It contains the following topics:

For more information about how to create and execute an RSP, see the Mainframe Connect Server Options *Programmer's Reference for Remote Stored Procedures*.

Creating RSPs

An RSP is a customer-written CICS program that resides on the mainframe. You can write an RSP in any one of the following languages:

- assembler
- COBOL II
- PL/1
- C

An RSP can use any CICS resource or access any CICS data source.

In an RSP, the DB2 access service recognizes argument values that are enclosed in either single or double quotes. Quoted argument values must be in the proper case. The entire statement has a 32KB size limit.

Executing RSPs

A client application executes an RSP by issuing a command appropriate for the SQL transformation mode in effect. Then, the DB2 access service passes this request to the MainframeConnect for DB2 access module, AMD2, which performs a CICS link to the specified RSP. If it does not find a procedure by that name, it searches the host request library (see “Executing host-resident requests” on page 219 for more information). The DB2 access service returns all results generated by the RSP to the client application.

In passthrough mode

A client application executes an RSP by issuing a use procedure command as follows:

use procedure [with [binary] data] *procedurename* [*arglist*];
[*input data*]

where:

- *procedurename* is the name of the RSP.
- *arglist* is the list of arguments.
- *input data* is the binary data.

A client application can send input data only when the SQL transformation mode is passthrough mode (or TSQL0 or TSQL1 backward compatible modes).

with binary data clause

If the with binary data clause is specified, you must configure the DB2 access service DefaultClientCodeset and DefaultTargetCodeset properties to unequal values. For example:

```
{ACS Required}
 DefaultClientCodeset=850
 DefaultTargetCodeset=500
```

These unequal property value settings enable the DB2 access service to recognize that all data between the semicolon and the end of the buffer is binary.

If the property values are equal, the DB2 access service performs ASCII and EBCDIC translation with the data records.

with data clause

If the with data clause is specified, the DB2 access service assumes that each line following the semicolon (;) is a data record. A single line with a semicolon follows the last record. The DB2 access service performs ASCII and EBCDIC translation with the data records.

Examples

The following examples show how to execute an RSP in passthrough mode:

```
use procedure bob &ARG1=red &ARG2=blue
```

```
use procedure bob this is variable text
```

Note Do not confuse the use procedure syntax with a T-SQL use command, which specifies an Adaptive Server database context.

In sybase mode

A client application executes an RSP by issuing an execute command as follows:

exec *procedurename* [*arglist*];

execute *procedurename* [*arglist*];

Examples

Following are examples of executing an RSP in sybase mode:

```
exec bob @arg1=red, @arg2=blue
```

```
execute bob this is variable text
```

The client application cannot send input pipes when in sybase mode (or TSQL2 backward-compatible mode).

Note RSPs do not support IXF records.

Creating host-resident requests

The DB2 access service supports host-resident requests, which are SQL statements stored in a special DB2 table called the host request library. A client application executes a host-resident request the same way it executes an RSP.

Rather than the client application issuing the SQL directly, you can define the statement and then have the client application execute the statement as a procedure. A host-resident request gives you more control over the SQL that a client application generates.

To add a request to the host request library, a client application executes a create request or a create procedure statement that contains the request name and the SQL text. The user who issues the statement must have all necessary privileges on the host request library table.

The following restrictions apply to a host-resident request:

- It must be a single SQL statement.
- Its name must be eight characters or less.
- Its maximum length is 32,000 characters.

In passthrough mode

When in passthrough mode (or TSQL0 or TSQL1 backward-compatible mode), use the following syntax to create a request:

create request *requestname*
 request command

For example:

```
create request selauth select * from authors
```

In sybase mode

When in sybase mode (or TSQL2 backward-compatible mode), use the following syntax to create a request:

create procedure *requestname* as
 request command

For example:

```
create procedure selauth as select * from authors
```

The DB2 access service does not support the following create procedure features of Transact-SQL™:

- Numbered procedures
- Default parameter argument values
- The with recompile option

The first two cause errors; the third is ignored.

Executing host-resident requests

A client application executes a host-resident request by issuing a SQL command appropriate for the SQL transformation mode in effect. The DB2 access service passes this request to the MainframeConnect for DB2 access module, AMD2, which retrieves the SQL command and executes it against DB2. The DB2 access service returns all results to the client application.

In passthrough mode

When in passthrough mode (or TSQL0 or TSQL1 backward compatible mode), use the following syntax to execute a host-resident request:

use request *requestname*

where *requestname* is the name of the host-resident request.

Some use request statements can be nested: One request can point to a second use request statement, and so on.

Note Do not confuse the use request syntax with a T-SQL use command, which specifies an Adaptive Server database context.

In sybase mode

When in sybase mode (or TSQL2 backward-compatible mode), execute a host-resident request by using the execute command as follows:

```
execute requestname
```

Variables in host-resident requests

Host-resident requests can contain up to 50 variables with names of any length. In passthrough mode, an ampersand (&) precedes each variable. In sybase mode, an (@) (at) symbol precedes each variable. Variables are specified as follows in passthrough and sybase mode:

passthrough mode:

&var1=*value1* &var2=*value2* ...

sybase mode:

@var1=*value1,* @var2=*value2*

Passthrough mode example

The following example shows how to create and execute a host-resident request with variables when in passthrough mode:

```
create request seltitle select * from titles
 where type = &vartitle and total_sales > &varsales
use request seltitle &vartitle=’psychology’ &varsales=2000
```

Sybase mode example

The following example shows how to create and execute a host-resident request with variables when in sybase mode:

```
create procedure seltitle as select * from titles
 where type = @vartitle and total_sales > @varsales
execute seltitle @vartitle=’psychology’, @varsales=2000
```

Deleting host-resident requests

To delete a request, use the appropriate command for the SQL transformation mode in effect.

In passthrough mode

When in passthrough mode (or TSQL0 or TSQL1 backward-compatible mode), delete a request by using the drop request command as follows:

```
drop request requestname
```

where *requestname* is the name of the host-resident request.

In sybase mode

When in sybase mode (or TSQL2 backward-compatible mode), delete a request by using the drop procedure command as follows:

```
drop procedure requestname
```

where *requestname* is the name of the host-resident request.

APPENDIX A

Configuration References and Code Set Tables

This appendix contains quick reference tables for configuration properties and code sets for translation.

This appendix contains the following topics:

Configuration properties

For detailed explanations of configuration properties, see Chapter 2, "Creating and Configuring DB2 Access Services."

The following table lists DB2 Access Service Library and DB2 access service properties in alphabetical order. The property category represents the subsection heading in the service library configuration file.

Table A-1: Configuration properties

Property name	**Property values**	**Property category {subsection name}**	**Global variable (GV) or set statement (SS)**
Allocate	[connect | request]	{Target Interaction}	GV and SS
APPCSecurity	[pgm | none | same]	{Target Interaction}	none
ApplicationValidationFile	*pathfilename*	{Client Interaction}	none
BinaryResults	[binary | char]	{Datatype Conversion}	GV and SS
CharConvertError	[reject | truncate]	{Data Conversion Errors}	GV and SS
ClientDecimalSeparator	*char*	{Client Interaction}	GV and SS
ClientIdleTimeout	*integer*	{Client Interaction}	none
ConnectionProtocol	[lu62 | tcpip]	{ACS Required}	none
ConnectionSpec1	*char*	{ACS Required}	none

Property name	**Property values**	**Property category {subsection name}**	**Global variable (GV) or set statement (SS)**
ConnectionSpec2	*char*	{ACS Required}	none
ConnectionSpec3	*char*	{ACS Required}	none
CSPCatalogQualifier	*qualifier*	{Catalog Stored Procedures}	GV and SS
CSPDBName	*dbname*	{Catalog Stored Procedures}	GV and SS
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
CSPQualByDBName	[no | yes]	{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_odbc]	{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
DefaultClientCodeset	*clientcodeset*	{ACS Required}	GV and SS
DefaultDate	*yyyy-mm-dd*	{Data Conversion Errors}	none
DefaultNum	*integer*	{Data Conversion Errors}	none
DefaultTargetCodeset	*targetcodeset*	{ACS Required}	GV
DefaultTime	*hh.mm.ss*	{Data Conversion Errors}	none
EnableAtStartup	[no | yes]	{Client Interaction}	none
FloatResults	[float | real | char]	{Datatype Conversion}	GV and SS
GatewayCompatible	[no | yes]	{Client Interaction}	GV
GraphicResults	[binary | char]	{Datatype Conversion}	GV and SS
Int2Results	[smallint | char]	{Datatype Conversion}	GV and SS
Int4Results	[int | char]	{Datatype Conversion}	GV and SS
LogConnectionStatistics	[no | yes]	{Logging}	none

Property name	Property values	Property category {subsection name}	Global variable (GV) or set statement (SS)
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
PasswordRequired	[no | yes]	{Target Interaction}	none
QuotedStringDelimiter	*char*	{Target Interaction}	none
RealResults	[float | real | char]	{Datatype Conversion}	GV and SS
SendWarningMessages	[no | yes]	{Client Interaction}	GV and SS
ServiceDescription	*char*	{Client Interaction}	GV
SvclibDescription	*char*	{Client Interaction}	GV
SQLInformationFile	*filename*	{Catalog Stored Procedures}	none
SQLTransformation	[passthrough | sybase | tsql0 | tsql1 | tsql2 | db2]	{Target Interaction}	GV and SS
StopCondition	[error | none | warning]	{Target Interaction}	GV and SS
StripBinaryZero	[no | yes]	{Client Interaction}	GV and SS
TargetDebug	[none | statistics | time | trace]	{Target Interaction}	GV and SS
TargetDecimalSeparator	*char* (default is a period)	{Target Interaction}	GV
TargetHasMixedData	[no | yes]	{Target Interaction}	none
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
TPName	*tpname*	{ACS Required}	none
TraceEvents	[no | yes]	{Tracing}	none
TraceHostCom	[no | yes]	{Tracing}	none
TraceInterface	[no | yes]	{Tracing}	none
TraceTarget	[no | yes]	{Tracing}	none

Property name	**Property values**	**Property category {subsection name}**	**Global variable (GV) or set statement (SS)**
TransactionMode	[short | long]	{Client Interaction}	GV and SS
TransferBatch	*integer*	{Transfer}	GV and SS
TransferErrorCount	*integer*	{Transfer}	GV and SS
Version	*versionstring*	{Client Interaction}	GV

Code set reference tables

This section contains code set identifiers that are used with two configuration properties:

- DefaultClientCodeset
- DefaultTargetCodeset

For a complete list of the code sets that can be used, see the *Unilib® Reference Manual Developer's Kit for Unicode*.

Values for DefaultClientCodeset

The following table shows code set values that you can use with the DefaultClientCodeset configuration property.

Table A-2: Values for DefaultClientCodeset configuration property

Code set identifier	**Description**
9	EUC-CNS encoding
10	EUC-GB encoding
11	Microsoft CP932
12	ISO 8859-2 Latin-2 Eastern Europe
13	ISO-885905 Latin/Cyrillic
14	ISO 8859-6 Latin/Arabic
15	ISO 8859-7 Latin/Greek
16	ISO 8859-8 Latin/Hebrew
17	ISO 8859-9 Latin-5 Turkish
26	Macintosh Cyrillic
27	Macintosh Eastern Europe
28	Macintosh Greek
29	Macintosh Turkish
30	HP Greek
31	HP Turkish
32	KOI8 - Cyrillic
33	Thai Standard
420	Arabic Bilingual
437	U. S. code page
819	ISO 8859-1 Latin
850	European code page
852	PC Eastern European
855	PC Cyrillic
857	PC Turkish
860	PC Portuguese
861	Icelandic
863	PC Canadian French code page
864	PC Arabic
866	PC Russian
869	PC Greek
874	Microsoft Thai SB code page
932	Japanese IBM J-DBCS
949	PC MS Korean
950	PC MS Traditional Chinese
954	EUCJIS

Code set identifier	Description
1250	MS Windows East Europe
1251	MS Windows Cyrillic
1252	MS Windows US
1253	MS Windows Greek
1254	MS Windows Turkish
1255	MS Windows Hebrew
1256	MS Windows Arabic
1257	MS Windows Baltic
1258	MS Windows Vietnamese
deckanji	Digital UNIX JIS encoding
eucjis	EUC-JIS encoding
mac	Standard Macintosh Roman
roman8	HP Roman8
sjis	Shift-JIS proper

Values for DefaultTargetCodeset

The following table shows all values that you can use with the DefaultTargetCodeset configuration property.

Table A-3: Values for DefaultTargetCodeset configuration property

Code set identifier	Description (alternate identifier)
37	IBM EBCDIC code page
273	IBM EBCDIC Germany/Austria code page
277	IBM EBCDIC Denmark/Norway code page
278	IBM EBCDIC Finland/Sweden code page
280	IBM EBCDIC Italian code
284	IBM EBCDIC Spain/Latin America code page
285	IBM EBCDIC U.K. code page
297	IBM EBCDIC France code page
420	Arabic Bilingual
500	Western Europe code page
870	Eastern Europe
874	Microsoft Thai SB code page
875	Greek code page

Code set identifier	**Description (alternate identifier)**
933 - *833* - *834*	MBCS Korean - SBCS - DBCS
935 - *836* - *837*	MBCS Simplified Chinese - SBCS - DBCS
936	PC Simplified Chinese
937 - *37* - *835*	MBCS Traditional Chinese - SBCS - DBCS
1047	MVS Open Edition
5026 - *290* - *4396*	MBCS IBM Kanji (Katakana) - SBCS - DBCS
5035 - *1027* - *4396*	MBCS IBM Kanji (Latin) - SBCS - DBCS
cxgwebc	MDI Gateway-compatible
cxpsebc	PeopleSoft Custom

APPENDIX B

Compatibility with MDI Database Gateways and Net-Gateway

This appendix provides information regarding the compatibility between DirectConnect and both the MDI Database Gateways and Net-Gateway.

This appendix contains the following topics:

Compatibility with MDI Database Gateways

If you have client applications for MDI Database Gateways, you can still use those applications with DB2 access services.

Compatibility with Net-Gateway

If you have client applications for Net-Gateway, those applications will work with TRS. Simply follow the file-moving instructions in the next subsection.

Table B-1: Moving Net-Gateway files to TRS files

Move Net-Gateway files	To these TRS files
For UNIX: $SYBASE/ngcid.<msg_server_name>	$SYBASE/$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) /*servername*/cfg/ngcid. <trs_service_library_name>
For TCP/IP on non-UNIX: %SYBASE%\ngcid.<msg_server_name>	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) \servername\cfg\<trs>.ngcid
For LU 6.2 on non-UNIX: %SYBASE%\<msg_server_name>.cid	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) \servername\cfg\<trs>.cid
For UNIX: $SYBASE/ngreg.<msg_server_name>	$SYBASE/$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) /servername/cfg/ngreg.<trs_service_library_name>
For TCP/IP on non-UNIX: %SYBASE%\ngreg.<msg_server_\name>	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) *servername*\cfg\<trs>.ngreg
For LU 6.2 on non-UNIX: %SYBASE%\<msg_server_name>.reg	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) *servername*\cfg\<trs>.reg
For UNIX: $SYBASE/nggrp.<msg_server_name>	$SYBASE/$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) /servername/cfg/nggrp.<trs_service_library_name>
For TCP/IP on non-UNIX: %SYBASE%\nggrp.<msg_server_name>	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) *servername*\cfg\<trs>.nggrp
For LU 6.2 on non-UNIX: %SYBASE%\<msg_server_name>.grp	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) *servername*\cfg\<trs>.grp

Move Net-Gateway files	**To these TRS files**
For UNIX: $SYBASE/ngrpc.<msg_server_name>	$SYBASE/$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) /<*servername*>/cfg/ngrpc.<trs_service_library_ name>
For TCP/IP on non-UNIX: %SYBASE%\ngrpc.<msg_server_name>	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) \servername\cfg\<trs>.ngrpc
For LU 6.2 on non-UNIX: %SYBASE%\<msg_server_name>.rpc	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) \servername\cfg\<trs>.rpc
For UNIX: $SYBASE/ngact.<msg_server_name>	$SYBASE/$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) /*servername*/log/ngact.<trs_service_library_name>
For TCP/IP on non-UNIX: %SYBASE%\ngact.<msg_server_name>	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) \servername\cfg\<trs>.ngact
For LU 6.2 on non-UNIX: %SYBASE%\<msg_server_name>.act	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT)$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) \servername\cfg\<trs>.act
For UNIX: $SYBASE/ngtds.<msg_server_name>	$SYBASE/$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) /*servername*/log/ngtds.<trs_service_library_name>
For TCP/IP on non-UNIX: %SYBASE%\ngtds.<msg_server_name>	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) \servername\cfg\<trs>.ngtds
For LU 6.2 on non-UNIX: %SYBASE%\<msg_server_name>.tds	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) \servername\cfg\<trs>.tds

Move Net-Gateway files	To these TRS files
For UNIX: $SYBASE/nglog.<msg_server_name>	$SYBASE/$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) /*servername*/log/ *servername*.log
For TCP/IP on non-UNIX: %SYBASE%\nglog.<msg_server_name>	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) *servername*\log\ <dcon_server_name>.log
For LU 6.2 on non-UNIX: %SYBASE%\<msg_server_name>.log	%SYBASE%\$SYBASE_ECON (UNIX) or %SYBASE_ECON% (NT) *servername*\log\ <dcon_server_name>.log

APPENDIX C

Using Sybase Mode Commands

This chapter describes the T-SQL syntax subset recognized in the Sybase SQL transformation mode. SQL commands, clauses, and other syntactical elements are presented in alphabetical order.

This chapter contains the following topics:

T-SQL commands

Many of the commands use up to three-part table names. DB2 supports the following three-part naming convention:

- *location_name* (database system name, current server)
- *authorization_ID* (owner)
- *table_name* or *view_name*

Note All sybase mode commands are issued as language commands, unless otherwise noted.

The following table lists each command and its description.

Table C-1: Transact-SQL commands

SQL command	Description
alter table	Adds new columns to an existing table.
begin transaction	Marks the starting point of a user-specified 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 (cursor command)	Removes rows from a table using a *cursor* command.
delete (dynamic command)	Removes rows from a table using a *dynamic* command.
delete (language command)	Removes rows from a table using a *language* command.
drop index	Removes an index from a table.
drop table	Removes a table.
drop view	Removes a view.
execute	Runs a system procedure or user-defined storage procedure.
grant	Assigns authorization to users.
insert (dynamic command)	Adds new rows to a table or view using a *dynamic* command.
insert (language command)	Adds new rows to a table or view using a *language* command.
prepare transaction	Checks to see if connections to databases are active.
revoke	Revokes authorization of users.
rollback transaction	Rolls back or aborts the current transaction.
select	Retrieves rows from the database objects.
truncate table	Truncates the table by removing all rows. This statement is not logged and is not part of any transaction.
update (cursor command)	Changes data in row made current by a read cursor (positional update).
update (dynamic command)	Changes data in existing rows using a *dynamic* command.
update (language command)	Changes data in existing rows using a *language* command.
use	Accesses an existing database.

For more information about the Sybase SQL transformation mode commands, see the *Sybase SQL Server Reference Manual*.

alter table

Description

Using a *language* command, alter table adds new columns to an existing table.

Syntax

alter table [*database*.[*owner*].]*table_name*

add *column_name datatype* {null | not null}

[{, *next_column*}...]

Parameters

table_name
is the name of the table to be changed.

column_name
is the name of a column to be added.

datatype
is the datatype of the column. Use only system datatypes, except bit. Certain datatypes expect a length, *n*, in parentheses:

```
datatype(n)
```

null | not null
specifies a null value if a user does not provide a value during an insertion and no default exists (for null), or that a user 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.

Examples

The following example adds the manager_name column to the publishers table. For each existing row in the table, a null value is assigned to the new column.

```
alter table publishers
add manager_name varchar(40) null
```

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

Description	A DB2 equivalent of this command does not exist. The DB2 access service sends both begin transaction and transaction_name to the target as begin transaction.
Syntax	none
Usage	Not defined.

commit transaction

Description
: Commits the work resulting from the current transaction.

Syntax
: commit transaction {*transaction_name*}

Parameters
: *transaction_name*
  is the name assigned to the transaction. It must conform to the rules for identifiers that are described in the *Sybase SQL Server Reference Manual*.

Examples

```
commit transaction
```

Usage

- The DB2 access service accepts this statement and translates it to the DB2 commit statement.
- If a transaction is not currently active, this statement has no effect.
- In sybase mode, transaction_name is stripped from the statement before it is passed on the target.
- The DB2 access service converts both commit transaction *transaction_name* and commit transaction without a transaction name to a commit statement.

create index

Description

Using a *language* command, create index adds a new index to an existing table.

Syntax

create [unique] index *index_name*
on [[*database.*]*owner.*]*table_name*
(*column_name* [, *column_name*]...)

Parameters

unique
: is an optional keyword that prohibits duplicate index values (also called key values).

index_name
: is the name of the index. Index names must be unique within a table but not 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)
```

Usage

- Columns of type bit, text, and image cannot be indexed.
- You cannot create an index on a view.
- The DB2 access service initially accepts the following T-SQL statement components but then strips them out of the statement:
  - clustered and nonclustered
  - *with { fillfactor = x, ignore_dup_key, sorted_data* }
  - *on segment_name*, which specifies the segment where the index is to be created

create table

Description

Creates a new table.

Syntax

create table [*database.*[*owner*].]*table_name* (*column_name datatype* {null | not null} [{, *next_column* }...]) [*on segment_name*]

Parameters

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 C-3 on page 242, several datatypes expect a length, *n*, in parentheses:

datatype(n)

null | not null
specifies a null value if a user does not provide a value during an insertion and no default exists (for null), or that a user 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.

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)
```

Usage

- T-SQL allows you to specify null or not null, with a default of not null. DB2 allows only not null to be specified, and the default is null. The following table shows how the DB2 access service transforms this clause.

Table C-2: Null transformations during T-SQL to DB2 create table

Transact-SQL specification	Transformed to...
null	<nothing>
not null	not null
<nothing>	not null

- The following table shows the DB2 access service transformation of datatype specifications.

Table C-3: Datatype conversions during T-SQL to DB2 create table

Transact-SQL datatype	DB2 datatype
tinyint	SMALLINT
smallint	SMALLINT
int	INT
numeric(p,s)	NUMERIC(p,s)
decimal(p,s)	DECIMAL(p,s)
float (double precision)	FLOAT
real	REAL
smallmoney	DECIMAL(10,4)
money	DECIMAL(19,4)
smalldatetime	TIMESTAMP
datetime	TIMESTAMP
char(n)	CHAR(*n)*
varchar(n)	VARCHAR(*n*)
text	LONG VARCHAR(*n*)
binary(n)	CHAR(*n*) FOR BIT DATA
varbinary(n)	VARCHAR(*n*) FOR BIT DATA
image	LONG VARCHAR *(n*) FOR BIT DATA
bit	SMALLINT

create view

Description

Creates a new view.

Syntax

create view [*database_name.]*[*owner*.]*view_name*
[(*column_name* [, *column_name*]...)]
as select [distinct] *select_statement*
[with check option]

Parameters

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.

as select
begins the select statement that defines the view.

distinct
specifies that the view cannot contain duplicate rows (optional).

select_statement
completes the select statement that defines the view. It can include more than one table and other views.

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.

Examples

The following example creates the new view from old view.

```
create view new_view (col1, col2)
 as select col1, col2 from old_view
```

Usage

You can use views as security mechanisms by granting authorization on a view but not on its underlying tables.

delete (cursor command)

Description

Using a *cursor* command, delete removes a row from a table. The row affected must have been made current by a *read* cursor.

Syntax

1. delete [[*database*.]*owner*.]{*table_name* | *view_name*}
2. delete [from] [[*database*.]*owner*.]{*table_name*|*view_name*}
   where current of *cursor_name*

Note Any valid object in the catalog can be substituted for *table_name* or *view_name*.

Parameters

from (after delete)
: 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 current of
: is a standard where clause.

Examples

```
declare c1 cursor for
 select * from tablea for update of col1
delete tablea where current of c1
```

Usage

The cursor can be reused multiple times before it is deallocated.

delete (dynamic command)

Description

Using a *dynamic* command, delete removes a row from a table.

Syntax

delete [[*database*.]*owner*.]{*table_name* | *view_name*}
 [where *search_conditions*]

Note Any valid object in the catalog can be substituted for *table_name* or *view_name*.

Parameters

where
: is a standard where clause.

search_conditions
: is a valid where clause component. It 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

```
delete from tablea
 where col1 = ”test”
```

Usage

- Following are relational operators that are supported in search conditions: =, <>, <, >, <=, >=, and LIKE.
- The prepared statement can execute multiple times before it is deallocated.

delete (language command)

Description

Using a *language* command, delete removes a row from a table.

Syntax

delete [from] [[*database.*]*owner*.]{*table_name*|*view_name*}
[where *search_conditions*]

delete [[*database*.]*owner*.]{*table_name* | *view_name*}
[from [[*database*.]*owner*.]{*table_name* | *view_name*}
[, [[*database*.]*owner*.]{*table_name* | *view_name*}]...]
[where *search_conditions*]

Parameters

from (after delete)
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.

from (after *table_name* or *view_name*)
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.

where
is a standard where clause.

search_conditions
is a valid where clause component. It 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.

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] T-SQL keyword parameter.

drop index

Description

Removes an index from a table in the current database.

Syntax

drop index *table_name.index_name*

The preceding syntax works *only* in sybase mode.

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 T-SQL, but ODBC supports only a single name per statement. If multiple names are present, multiple ODBC statements are generated.
- ASE/CIS passes the drop index command to DirectConnect for z/OS as a language event.
- To get information about existing indexes on a table, use:

  sp_helpindex *table_name*

drop table

Description

Removes a table definition and all of its data, indexes, triggers, and authorization specifications from the database.

Syntax

drop table [[*database*.]*owner*.]*table_name*

Parameters

table_name
is the name of the table to be dropped.

Examples

```
drop table authors
```

Usage

- T-SQL allows you to drop multiple tables in one statement, but ODBC does not. If multiple tables are encountered, transformation generates multiple statements.
- ODBC allows the keywords CASCADE and RESTRICT to be used in the statement. Since T-SQL does not support this, the keywords are not generated during transformation.
- ASE/CIS passes the drop table command to DirectConnect for z/OS as a language event.

drop view

Description
Removes one or more views from the database.

Syntax
drop view [*databasename*], [*owner*].*view*

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
```

Usage
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

Description
Assigns authorizations to users.

Syntax
To grant authorization to access database objects:

grant {all [privileges]| *permission_list*}
on {*table_name* | *view_name*} | *stored_procedure_name*}
to {public | *name_list*} with grant option]

To grant authorization to create database objects:

grant {*all [privileges*] | *command_list*}
 to {*public* | *name_list*}

Note Any valid object in the catalog can be substituted for *table_name* or *view_name*.

Parameters

all
: when used to assign authorization to access database objects (first syntax format), all specifies that all privileges applicable to the specified object are granted or revoked.

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.

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.

public
: is all users of the "public" group, which includes all users of the system.

name_list
: is a list of users' database names or group names or both, separated by commas.

with grant option
: allows the users specified in *name_list* to grant the privileges specified by *permission_list* to other users.

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 all on titles
 to public

grant all
 to public

grant update on authors
 to mary
 with grant option

grant select, update on titles(price)
 to bob
 with grant option
```

Usage

- DB2 does not allow you to grant authorization to a stored procedure.
- You can substitute the word from for to in the grant syntax.
- You can only grant or revoke authorizations on objects in the current database.
- *Role_name*, which allows you to grant authorizations to all users who have been granted a specific role, is not supported. However, if you include it in the command, an error does not occur.

insert (dynamic command)

Description

Using a *dynamic* command, insert adds a new row to a table or view.

Syntax

insert [*database*.[*owner*.]]{*table_name*|*view_name*}[(*column_list*)]
values (*? [, ?*]...)

Note Any valid object in the catalog can be substituted for *table_name* or *view_name*.

Parameters

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 a select clause) is in the same order.

values
is a keyword that introduces a list of expressions.

? (question mark)
specifies a parameter marker passed by the application.

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 for z/OS as the following series of dynamic SQL commands:

  prepare

  execute

  close

  deallocate
- The values in the values list are passed as dynamic SQL parameters.

insert (language command)

Description

Using a *language* command, insert adds a new row to a table or view.

Syntax

insert [into] *database*.[*owner*.]{*table_name*|*view_name*}[(*column_list*)]
 values (*value1, [,value2]...*)

Note Any valid object in the catalog can be substituted for *table_name* or *view_name*.

Parameters

into
is optional.

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 a select clause) is in the same order.

values
is a keyword that introduces a list of expressions.

Examples

```
insert titles (title_id, title, type, pub_id, notes,
pubdate,contract)
 values (docid, docno, docdate)
```

Usage Not defined.

prepare transaction

Description Determines whether a server is still connected.

Syntax prepare transaction

(not supported)

Examples None

Usage Not supported.

revoke

Description Revokes authorizations from users.

Syntax

revoke [grant option for]{all [privileges]| *permission_list*}
on {*table_name* | *view_name*} | *stored_procedure_name*}
from {public | *name_list*} [cascade] revoke {*all* [*privileges*] | *command_list*}
from {public | *name_list*}

Parameters

all
(in the first syntax format) specifies that all privileges applicable to the specified object are revoked when used to revoke authorizations to access database objects.
The second syntax format can revoke create command authorizations:

- When the System Administrator uses this command, all create authorizations are revoked.
- When the database owner uses this command, all create authorizations are revoked except create database.

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 the specified table in the database.

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.

public
is all users of the "public" group, which includes all users of the system.

name_list
is a list of users' database names and group names, separated by commas.

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 can happen even if it is not specified by the user. For example, UserA has granted UserB some privileges, and in turn, UserB granted some to UserC. If UserA is revoked, all privileges that UserA granted to UserB and UserB indirectly granted to UserC are revoked.

Note Consult the *IBM DB2 SQL Reference* manual to see how IBM implements DB2 authorization schemes.

Examples

```
revoke insert, delete
 on titles
 from mary, sales
```

```
revoke all on titles
 from public
```

Usage

- DB2 does not support the revoking of a stored procedure.
- Authorizations can only be revoked on objects in the current database.

- grant and revoke commands are order-sensitive. When a conflict occurs, the most recently issued command takes effect.
- The word to can be substituted for the word from in the revoke syntax.
- The DB2 access service does not support *role_name*.

rollback transaction

Description
: Rolls back a user-specified transaction to the beginning of the transaction.

Syntax
: rollback {transaction | tran | work}
 [*transaction_name*]

Parameters
: tran
  is another term for transaction.

  work
  is another term for transaction.

  transaction_name
  is the name assigned to the transaction. It must conform to the rules for identifiers.

Examples

```
rollback transaction
```

Usage

- In sybase mode, *transaction_name* is stripped from the statement before it is passed to the target.
- The DB2 access service converts rollback transaction, rollback work, and rollback tran statements to a rollback statement.

select

Description
: Retrieves rows from database objects. You can issue this command either as *language* command or a CT-Library *cursor* request.

Syntax
: select *select_list* [from [[*database*.]*owner*.]{*table_name* |*view_name*}
[,[[*database*.]*owner*.]{*table_name*|*view_name*}]...] [where *search_conditions*]

Parameters

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 the items listed previously

from
indicates the particular tables and views to use in the select statement.

table_name, *view_name*
lists tables and views used in the select statement.

If more than one table or view is in the list, their names are separated by commas. Table names and view names are given correlating names. This is done 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.

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. There is no limit on 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.

Examples

```
select count(*) from publishers for read only
```

```
select pub_id, pub_name, city, state from publishers for
read only
```

```
select pub_name, pub_id
 from publishers
```

```
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

- The TEXTPTR() function cannot appear in the select list.
- All Adaptive Server 10.x aggregate functions are supported:
    - sum (distinct)
    - avg (distinct)
    - count (distinct)
    - count (*)
    - max (expression)
    - min (expression)
- The DB2 access service does not transform correlation names.
- You can issue the select command either as a language command or a client-based cursor request.
- If a cursor is passed a new set of parameters before it is opened, it can be reused multiple times.
- If passed as a cursor command, the data values used in the where clause search conditions are passed as cursor parameters. These parameters use the datatype associated with the column.
- Cursor parameters are indicated with an @ (at) symbol.
- Refer to sp_capabilities for specific functions that the DB2 access service supports.

truncate table

Description

Removes all rows from a table.

Syntax

truncate table [[*database*.]*owner*.]*table_name*

Parameters

table_name
is the name of the table to be truncated.

Examples

```
truncate table authors
```

Usage

- The client application passes the truncate table command to the DB2 access service as a language command.
- In sybase mode, the DB2 access service converts truncate table to a delete command without a where clause.

update (cursor command)

Description

Changes data in a row made current by a read cursor command, either by adding data or by modifying existing data.

Syntax

update [[*database*.]*owner*.]{*table_name* | *view_name*}
set *column_name1 = @p1*][, *column_name2 = @p2*]...

Note Any valid object in the catalog can be substituted for *table_name* or *view_name*.

Parameters

set
specifies the column name and assigns the new value. The value passes as a cursor parameter.

Examples

```
update authors
 set au_lname = @p1
```

The row made current by the cursor *authors_cursor* is modified, and the column *au_lname* is set to the value of the parameter @p1.

Usage

- The cursor can be reused multiple times before it is deallocated.
- You can include a from clause in the update statement, but because DB2 does not support it, the DB2 access service ignores it.

update (dynamic command)

Description

Using a *dynamic* command, update changes data in existing rows of the referenced table.

Syntax

update [[*database*.]*owner*.]{*table_name* | *view_name*}
set *column_name1* = ? [, *column_name2 = ?]...*
[where *search_conditions*...]]

Parameters

set
specifies the column name and assigns the new value. The value passes as a parameter.

where
is a standard where clause.

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

```
update authors
 set au_lname = ?
 where au_id = ?
```

The au_lname column is set to the value of <*parameter 1*> (indicated by a "?"), where the value of *au_id* is equal to the value of <*parameter 2*> (indicated by the second "?").

Usage

- You can include a from clause in the update statement, but because DB2 does not support it, the DB2 access service ignores it.
- You can substitute *table_name* and *view_name* with any valid object in the catalog.
- The following relational operators are supported in search conditions: =, <>, <, >, <=, >=, and LIKE.

update (language command)

Description

Changes data in existing rows, either by adding data or by modifying existing data. Use this as a *language* command.

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*)}]...
[,[[*database*.]*owner*.]{*table_name*|*view_name*}]...] [where *search_conditions*]

Parameters

set
specifies the column name and assigns the new value. The value can be an expression or a null. When more than one column name and value pair are listed, they must be separated by commas.

where
is a standard where clause.

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

```
update authors
 set au_lname = "MacBadden"
 where au_lname = "McBadden"
```

Usage

- You can include a from clause in the update statement, but because DB2 does not support it, the DB2 access service ignores it.
- You cannot update views defined with the distinct clause. However, when the view is created, the select language command allows the term distinct to be used with it. For more information, see the *Sybase SQL Server Reference Manual.*

use

Description

Accesses an existing database.

Syntax

use *database_name*

Parameters

database_name
is the name of the database you want to access.

Examples

```
use authors
```

Usage

- The DB2 access service returns the database name as part of the result set in an sp_sqlgetinfo call. It is located in the SQL_DATABASE_NAME attribute.
- If you issue a use database_name command with more than 127 characters, DirectConnect stores only the first 127 characters that you enter on this command.

Glossary

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 component of DirectConnect. 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 Enterprise	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 **RPC**, **service library**.
advanced program-to-program communication (APPC)	Hardware and software that characterize the LU 6.2 architecture and its various implementations in products. See also **LU 6.2**.
application program interface (API)	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.
ASE/CIS	Adaptive Server Enterprise / Component Integration Services (formerly OmniConnect). A variation of Sybase SQL Server that provides a Transact-SQL interface to various sources of external data, including host data files and tables in other database systems. The name replaces the names "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**.
catalog stored procedure (CSP)	A procedure that provides information about tables, columns, and authorizations. It is used in SQL generation and application development.
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/server**.
Client-Library (CT-Library or CT-LIB)	A Sybase API that allows a client application to interact with Open Server applications.
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 access and management for multiple clients. See also **client application**.
Client Services Application (CSA)	A user-written CICS program, initiated on the host, that uses the Client Services for CICS API to invoke MainframeConnect for DB2 as a client to DirectConnect or to Adaptive Server. Compare with **Remote Stored Procedures (RSPs)**.
clustered index	An index in which the physical order and the logical (indexed) order is the same. The leaf level of a clustered index represents the data pages themselves. Compare with **non-clustered index**.
Coded Character Set Identifier (CCSID)	A number that uniquely identifies a specific set of encoding scheme identifier, character set identifier(s), code page identifier(s), and additional coding-related required information.
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**.
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	In programming languages, the transformation between values that represent the same data item but belong to different datatypes. Information can be lost due to conversion because accuracy of data representation varies among different data types. See **character set conversion**.
Customer Information Control System (CICS)	An IBM-licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.
database management system (DBMS)	A computer-based system for defining, creating, manipulating, controlling, managing, and using databases. The software for using a database can be part of the database management system, or it can be a stand-alone database system. Compare with **relational database management system (RDBMS)**.
data definition language (DDL)	A language for describing data and their relationships in a database.
datatype	A keyword that identifies the characteristics of stored information on a computer. Some common datatypes are char (character), int (integer), smallint (small integer), date, time, numeric, and float. Different databases support different datatypes.
DB-Library (DB-LIB)	A Sybase and Microsoft API that allows client applications to interact with ODS applications. See also **application program interface (API)**.
DB2 access service	The named set of properties, used with an DB2 access service library, to which clients connect. Each DirectConnect server can have multiple services.
DB2 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 DB2 access service library interacts with exactly one target and is named accordingly. See also **service library**.
DB2-SQL	IBM SAA (Systems Application Architecture) embedded SQL. The ANSI SQL-1-compliant SQL syntax used by DB2 UDB.
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** and **dynamic SQL**. Compare with **bulk copy transfer**.

DirectConnect	A Sybase Open Server application that provides access management for non-Sybase database copy management (transfer) and remote systems management. Each DirectConnect consists of a server and one or more service libraries and provides access to a specific data source. DirectConnect replaces the “MDI Database Gateway” and “OmniSQL Access Module” names and products. Compare with **Enterprise Connect**.
DirectConnect Manager	A Sybase Windows application that provides remote management capabilities for DirectConnect products. These capabilities include starting, stopping, creating, and copying services.
DirectConnect for z/OS	A Sybase LAN-based solution that communicates with 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 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 server	The component that provides general management and support functions (such as log file management) to service libraries.
DirectConnect DB2 access service	The named set of properties, used with a DirectConnect DB2 Access Service Library, to which clients connect.
DirectConnect Access Service Library	The component that provides a set of functions within the DirectConnect server environment.
dynamic link library (DLL)	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	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 being coded directly into the application program. The SQL statement can change several times while the program runs. Compare with **static SQL**. Compare with **destination-template transfer**.
embedded SQL (ESQL)	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	The encompassing architecture upon which Sybase implements heterogeneous data integration. Compare with **DirectConnect**.

graphical user interface (GUI)	A type of computer interface consisting of a visual metaphor of a real-world scene, often of a desktop. Within that scene are icons, representing actual objects, that the user can access and manipulate with a pointing device.
InstallShield	The Sybase DirectConnect server installation program. An interactive program that is used to install DirectConnect products and programs that support them.
interfaces file	An operating system file that determines how the host client software connects to a Sybase product. The file must be available on each machine that connects to DirectConnect or other Sybase products.
keyword	A word or phrase that is reserved for exclusive use by Transact-SQL. Also known as a reserved word.
logical unit (LU)	A type of network-accessible unit that enables end users to gain access to network resources and communicate with each other. See also **end user**.
logical unit 6.2 (LU6.2)	A type of logical unit that supports general communication between programs in a distributed processing environment. See also **APPC**.
Mainframe Access Product (MAP)	Sybase products that enable desktop computer users to communicate with mainframes in a client/server environment.
Mainframe Connect	A Sybase suite of products that provide access to mainframe data.
MainframeConnect for DB2 UDB	A Sybase mainframe solution that provides dynamic access to DB2 data. It replaces the OmniSQL Access Module for DB2 and the functionality of the MDI Access Server.
MDI Database Gateway	An MDI legacy product that gives client applications access to supported data sources, such as DB2.
MVS (Multiple Virtual Storage)	An IBM operating system that runs on most System/370 and System/390 mainframes. It supports 24-bit addressing up to 16 megabytes.
Net-Gateway	A Sybase product that provides communication between the mainframe and the LAN server.
Net-Library (Net-Lib)	A Sybase product that lets PC applications become clients of SQL Server or Open Server. See also **client**, **Open Server**, **SQL Server**.
network protocol	A set of rules governing the way computers communicate on a network.
non-clustered index	An index that stores key values and pointers to data. The leaf level points to data pages rather than containing the data itself. Compare with **clustered index**.

null	Not having an explicitly assigned value.
OmniConnect	Functionality incorporated into CIS of Adaptive Server is referred to as ASE/CIS. See **ASE/CIS**.
Open Client	A Sybase product 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 Client application	An application written using Open Client libraries.
Open ClientConnect	A Sybase product that provides capability for the mainframe to act as a client to LAN-based resources.
Open Database Connectivity (ODBC)	A Microsoft API that allows access to both relational and non-relational databases. ODBC allows client application developers to produce vendor-neutral Windows applications that can access data sources without including code for a specific database. See also **application program interface (API)**.
Open Server	A Sybase and Microsoft product that provides the tools and interfaces required to create a custom server.
Open Server application	A custom server application built with Sybase Open Server.
Open ServerConnect	A Sybase product that provides capability for programmatic access to mainframe data.
Partner Logical Units (PLU)	The LU on the remote machine in a conversation. See also **LU** and **conversation**.
precision minus scale	Scale is the number of digits to the right of the decimal; precision is the total number of digits. Precision minus scale is the number of digits to the left of the decimal. For example, using the number 123.45, scale is 2, precision is 5, and precision minus scale is 3.
primary database	In transfer processing, the database accessed by the DB2 access service. For example, for DirectConnect for z/OS, the primary database is DB2. Compare with **secondary database.**
property	A setting for a server or service that defines the characteristics of the service, such as how events are logged or how datatypes are converted. Compare with **parameter** and **value**.

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 (relops)	Operators supported in search conditions. Examples are: =, <>, <, >, <=, >=, and LIKE.
remote procedure call (RPC)	A stored procedure executed on a different server from the one onto which a user is logged.
remote stored procedure (RSP)	A customer-written CICS program that resides on the mainframe and communicates with MainframeConnect for DB2 UDB. See also **CICS**, **stored procedures**.
remote systems management	A feature that allows a Systems Administrator to manage multiple DirectConnect servers and multiple services from a client. See **DirectConnect Manager**.
request	One or more database operations the application sends as one unit to the database. During a request, the application gives up control to the DBMS and waits for its response. Depending on the response, the application commits or rolls back the request. One or more requests can be grouped into a single unit of work. See also **commit**, **rollback**, **unit of work**.
rollback	An instruction to a database to not implement the changes requested in a unit of work and to return to the pre-transaction state. Compare with **commit**. See also **transaction** and **unit of work**.
scale	The maximum number of digits that can be stored to the right of the decimal point by a numeric or decimal datatype. The scale must be less than or equal to the precision.
secondary connection	The connection specified in the transfer statement. It represents anything that can be accessed using Open Client Connect, such as Adaptive Server or another DB2 access service.
secondary database	When performing transfers, one of the following: the Adaptive Server, another DirectConnect service, an MDI Database Gateway, or another 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. Compare with **client**. See also **client/server**.

server process ID (SPID)	A positive integer that uniquely identifies a client connection to the server. In Open Server, SPID is a thread ID. Each connection has its own thread. A client can have multiple connections to the server, each with its own SPID.
service	A functionality available to DirectConnect applications. It is essentially the pairing of a service library and a set of specific configuration properties.
service library	For DirectConnect, a set of configuration properties that determine service functionality. Examples of service libraries include DB2 access service libraries, transfer service libraries, administrative service libraries, and transaction router service libraries. See also **DB2 Access Service Library**, **Administrative Service Library**, **Transaction Router Service Library**.
service name redirection	A file for service name resolution that allows a Systems Administrator to create an alternative mechanism to map connections with services. See also **service name resolution**. Compare with **direct resolution**.
service name redirection file (SNRF)	A type of service name resolution that allows a Systems Administrator to create an alternative mechanism to map connections with services. See also **service name resolution**. Compare with **direct resolution**.
service name resolution	The mapping by the DirectConnect server of an incoming service name to an actual service. See also **direct resolution**, **service name redirection**.
structured query language (SQL)	An IBM industry-standard language for processing data in a relational database.
SQL descriptor area (SQLDA)	A set of variables that are used in the processing of certain SQL statements. The SQLDA is intended for dynamic SQL programs.
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. The file must be on every client machine that connects to SQL Servers.
SQL 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.
statement	Basic unit of SQL, a single SQL operation, such as select, update, or delete.

static SQL	SQL statements that are embedded within a program and are prepared during the program preparation process before the program runs. After being prepared, the statement itself does not change, although values of host variables specified by the statement can change. Compare with **call level interface** and **dynamic SQL**.
stored procedures	A collection of SQL statements and optional control-of-flow statements stored under a particular name. Sybase Adaptive Server stored procedures are called "system procedures." See also **remote stored procedure**, **system procedures**, **host-resident requests** and **catalog stored procedures**.
structured query language	An IBM industry-standard language for processing data in a relational database.
subsystem	A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system.
syntax	Rules for how to construct a statement.
system administrator	The user in charge of server system administration. For DirectConnect, the user responsible for installing and maintaining servers and service libraries.
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 unambiguously identified by means of one or more arguments.
Tabular Data Stream (TDS)	An application-level protocol that Sybase clients and servers use to communicate. It describes commands and results.
target	A system, program, or device that interprets, rejects or satisfies, and replies to requests received from a source.
target database	The database to which DirectConnect transfers data or performs operations on specific data.
trace	The process of recording the sequence in which the statements in a program execute and, optionally, the values of the program variables that the statements contain.
Transaction Router Service	The DirectConnect program that accepts requests from workstation-based clients and routes them to Open ServerConnect.
Transaction Router Service library	A service library that facilitates access to remote transactions, allowing customers to execute thousands of transactions from virtually any mainframe data source. See also **service library**.

Transact-SQL (T-SQL)	A Sybase-enhanced version of the SQL database language used to communicate with SQL Server. Enhancements include data integrity features and stored procedures.
transfer	A DirectConnect feature used to move data or copies of data from one database to another. See also **bulk copy transfer**, **destination-template transfer**.
unit of work	One or more database operations grouped under one 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**.
wildcard	Special character that represents a range of characters in a search pattern. For example, the percent sign (%) represents any string of zero or more characters.
z/OS	An IBM operating system that runs on z/OS mainframes.

Index