SYBASE®

Users Guide for Transaction Router Services

Mainframe Connect™ DirectConnect™ for z/OS Option

15.0

[Microsoft Windows, Linux, and UNIX]

DOCUMENT ID: DC38581-01-1500-01

LAST REVISED: August 2007

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

This guide describes how to configure, control, monitor, and use Transaction Router Service (TRS). TRS allows Sybase® clients to access data stored on a mainframe computer. TRS supports mainframe connections for LU 6.2 or TCP/IP networks.

Audience

This book is written for:

- Application Programmers, who develop organization-specific programs using the major features of Mainframe Connect™ DirectConnect™. for z/OS Option
- System Administrators, who install and test Mainframe Connect DirectConnect for z/OS Option. When Mainframe Connect DirectConnect for z/OS Option is running, System Administrators provide ongoing administration support, disaster recovery, and troubleshooting support.
- System Programmers, who install and test Mainframe Connect DirectConnect for z/OS Option. System Programmers also provide product administration, troubleshooting, and disaster recovery.

How to use this book

This guide describes a set of tasks, with each chapter representing a task. The following table shows how this book is organized.

Note Throughout this guide, the Mainframe Connect DirectConnect for z/OS Option will often be referred to as DirectConnect for z/OS Option.

See	When you are ready to.....
Chapter 1, “Introducing Transaction Router Service”	• Understand Enterprise Connect™ products • Understand how the DirectConnect for z/OS Option and TRS work together
Chapter 2, “Creating a TRS Library”	• Set up a TRS configuration file • Define the properties in that file • Use the file to establish a TRS
Chapter 3, “Configuring a TRS”	• Configure TRS • Perform administration procedures • Use the TRS elements that require administration • Review quick reference tables for the procedures
Chapter 4, “Accessing Catalog Information with CSPs”	Use CSPs to access the DB2 UDB catalog
Chapter 5, “Configuring a TRS Library for Security”	Control client access to TRS, to specific host connections, and to mainframe transactions
Chapter 6, “Using Password Expiration Management (PEM) with TRS”	Implement and use the Advanced Program-to-Program Communications (APPC) Password Expiration Management (PEM) function with TRS
Chapter 7, “Controlling a TRS”	Use the controlling administration tasks that TRS may need while it is running
Chapter 8, “Monitoring a TRS”	• Find information about TRS users, connections, regions, and Remote Procedure Calls (RPCs) • Find information about the trace status of TRS • Determine the options specified when TRS started
Appendix A, “Sending Requests to TRS”	For clients using TRS to access mainframe data
Appendix B, “Testing a TRS Installation with Sample Programs”	Test a TRS installation by running mainframe access products sample programs
Appendix C, “Localization”	Set up an application to run in a particular national language environment
Appendix D, “TRS Process User Exits”	Implement processing user exits

Note In this guide, Mainframe Connect DirectConnect for z/OS Option will be often referred to as “DirectConnect for z/OS Option.”

Related documents

To configure and administer DirectConnect for z/OS access services, use the Mainframe Connect DirectConnect for z/OS Option *Users Guide for DB2 Access Services*.

To install and administer mainframe products, use the following documents:

- Mainframe Connect DB2 UDB Option *Installation and Administration Guide* for CICS and IMS
- Mainframe Connect Client Option *Installation and Administration Guide* for CICS
- Mainframe Connect Client Option *Installation and Administration Guide* for IMS and MVS
- Mainframe Connect Server Option *Installation and Administration Guide* for IMS and MVS
- Mainframe Connect Server Option *Installation and Administration Guide* for CICS
- Open Client™ and Open Server™ *Common Libraries Reference Manual*
- Open Client *Client-Library/C Programmers 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 SyBooks™ CD, and the Sybase 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 SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks 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 Sybase 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 Click Certification Report.

3 In the Certification Report filter select a product, platform, and time frame and then click Go.

4 Click a Certification Report title to display the report.

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

1 Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.

2 Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.

3 Select Search to display the availability and certification report for the selection.

❖ **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

This type of information	Looks like this
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*]

This 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.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Mainframe Connect DirectConnect for z/OS Option and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

See a Section 508 compliance statement for Mainframe Connect DirectConnect for z/OS OptionVoluntary Product Assessment Templates at http://www.sybase.com/detail_list?id=52484.

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 Transaction Router Service

Topic	Page
DirectConnect for z/OS Option architecture	1
DirectConnect for z/OS Option components	2
DirectConnect for z/OS Option environment	5
Mainframe Connect options	7
How TRS differs from a DB2 access service	8
DirectConnect Manager	10

DirectConnect for z/OS Option architecture

Mainframe Connect DirectConnect for z/OS Option (hereafter referred to as DirectConnect for z/OS Option) is a local area network (LAN)-based middleware gateway and server that provides access to non-Sybase data and applications. DirectConnect for z/OS Option serves as a fundamental building block for highly-scalable database middleware applications.

DirectConnect for z/OS Option is Open Server-based software that supports DB-Library™ and Client-Library™ (CT-Library) application programming interfaces (APIs).

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

DirectConnect for z/OS Option 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 (TRS and DB2), which are the logical points of connection for DirectConnect for z/OS Option clients.

The following subsections describe each of these components.

DirectConnect for z/OS Option components

This section describes the DirectConnect for z/OS Option components:

- DirectConnect server
- DirectConnect service libraries
- DirectConnect access services

Note The DirectConnect for z/OS Option is one of four Mainframe Connect options. The other three are Client Option, DB2 UDB Option, and Server Option. For more information, see “Mainframe Connect options” on page 7.

DirectConnect server

The DirectConnect server provides the following management and support functions for DirectConnect service libraries:

- Routing client connections to the appropriate access service based on parameters in the *interfaces* or *sql.ini* file.
- Providing a single log file and a trace 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.

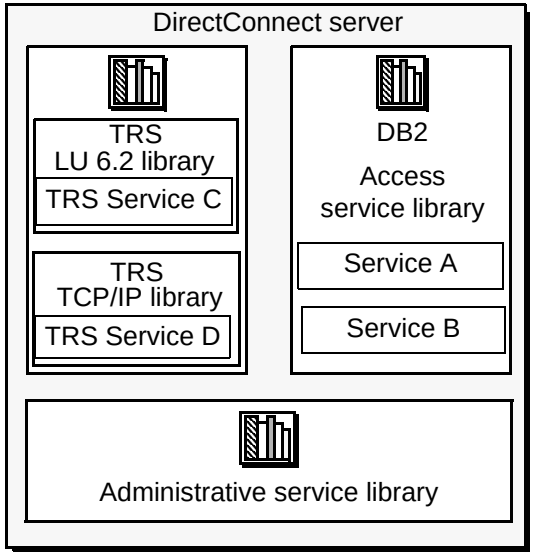
DirectConnect service libraries

Figure 1-1 shows a DirectConnect server, on which reside four service libraries. The first three libraries contain a set of configuration properties that describes how its access services will function:

- TRS library, which contains an LU 6.2 service library and a TCP/IP service library, each of which contains a set of configuration properties that determine how these TRS access services will function
- DB2 access service library, which contains a set of configuration properties that determine how DB2 access services will function
- Administrative service library, which contains software that provides specific administrative services for all DirectConnect libraries, such as writing to logs and allowing remote configuration of access services through DirectConnect Manager

  For more information, see “DirectConnect Manager” on page 10.

Figure 1-1: DirectConnect server, service libraries, and access services



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. The following sections describe the two types of access services.

DB2 access services

A DB2 access service works with the Mainframe Connect DB2 UDB Option for CICS and the Mainframe Connect DB2 UDB Option for IMS to allow clients to access DB2 UDB data. Each access service has a specific set of configuration properties that:

- Transforms SQL
- Converts datatypes
- Supports RPCs
- Transfers data between DB2 UDB and other servers accessible through Open Client
- Supports Catalog Stored Procedures (CSPs) and system stored procedures

- Supports remote stored procedures (RSPs) and host-resident requests

For more information, see the Mainframe Connect DirectConnect for z/OS Option *Users Guide for DB2 Access Services*.

Transaction Router Service (TRS)

Residing in the TRS library, a Transaction Router Service provides access to DB2 UDB data and supports Mainframe Connect Server Options mainframe applications, defined to TRS as remote procedure calls (RPCs).

TRS routes requests from remote LAN-based clients to Mainframe Connect Server Option transactions. In addition, it can route requests to the Mainframe Connect DB2 UDB Option and return results to the client.

As shown in Figure 1-1, there are two TRS service libraries residing on the DirectConnect server:

- *TRSLU62* service library, which contains access services that use the LU 6.2 communications protocol to talk to Mainframe Connect or any Mainframe Connect Server Option application running in CICS
- *TRSTCP* service library, which contain TRS access services that use the Transmission Control Protocol/Internet Protocol (TCP/IP) communications protocol to talk to the Mainframe Connect DB2 UDB Options or any Mainframe Connect Server Option 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.

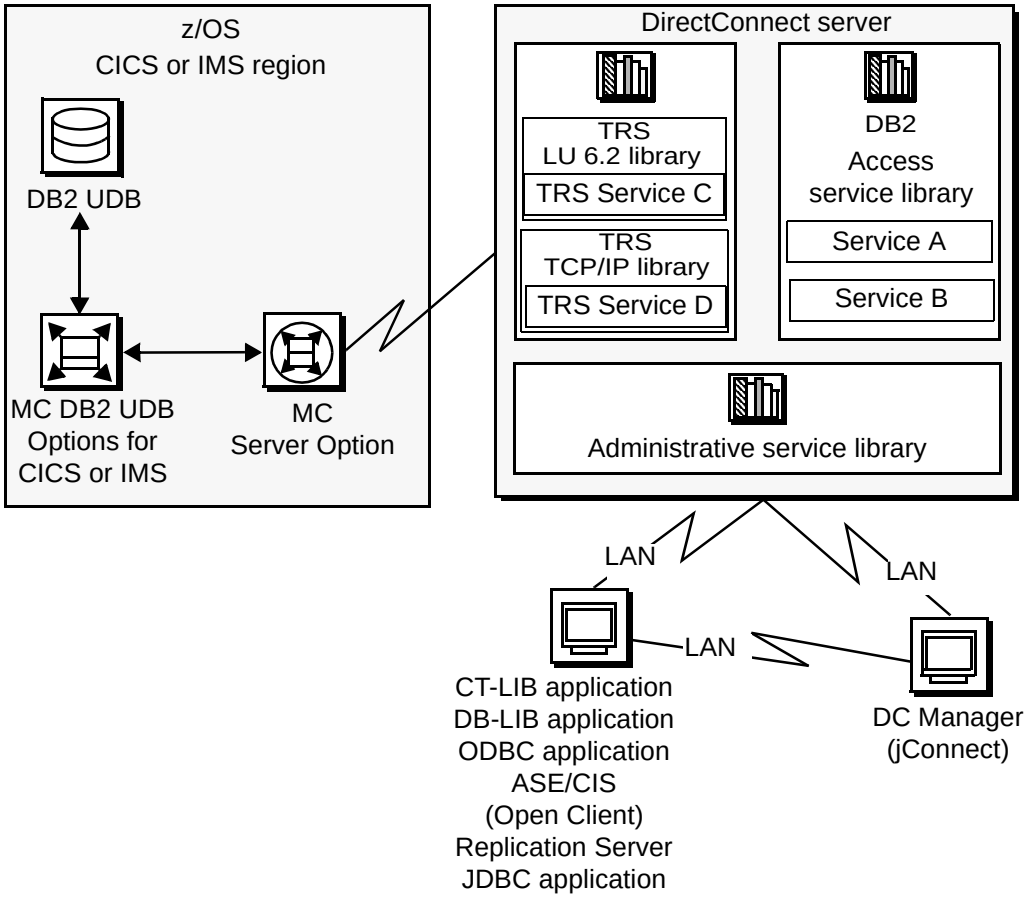
Security can also be configured on a transaction or user basis.

DirectConnect for z/OS Option environment

Figure 1-2 shows the relationship of the DB2 access service library and TRS library with various components of the client workstation, LAN, and mainframe environments.

Note Due to space constraints, Mainframe Connect is abbreviated to “MC.”

Figure 1-2: DirectConnect for z/OS Option environment



The process

As shown, the request from a client application goes over the LAN to 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 “Creating additional TRS configurations” on page 15.

Mainframe Connect options

This section describes options shown in Figure 1-2 that DirectConnect for z/OS Option interacts with to provide mainframe access for LAN client requests.

Mainframe Connect DB2 UDB Options

The Mainframe Connect DB2 UDB Option for CICS and and the Mainframe Connect DB2 UDB Option for IMS can work with Mainframe Connect DirectConnect for z/OS Option to provide access to mainframe data. They perform these 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 Option invokes the Mainframe Connect DB2 UDB Option to access mainframe data on behalf of its Open Client-based clients, such as:

- ASE/CIS
- ASE through RPCs
- Enterprise Application Server
- JDBC or ODBC applications
- Replication Server

Note The Mainframe Connect DB2 UDB Option for CICS and and the Mainframe Connect DB2 UDB Option for IMS are hereafter referred to as the DB2 UDB Options for CICS and for IMS.

Mainframe Connect Server Option

The Mainframe Connect Server Option is a programming environment that allows you to create mainframe transactions that are accessible to Sybase client applications. To provide this access, Mainframe Connect Server Option uses traditional Open Server APIs.

These transactions provide access to virtually any CICS and IMS 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 ASE/CIS, thus allowing results to be accessed or joined with data from other targets

LAN-side client applications access Mainframe Connect Server Option transactions directly through DirectConnect for z/OS Option or indirectly through ASE/CIS or a Sybase Adaptive Server RPC.

Mainframe Connect Client Option

The Mainframe Connect Client Option is a programming environment that allows you to create mainframe applications that access:

- LAN data residing on a Sybase Adaptive Server or other supported data sources
- Other CICS regions

To provide this access, the Mainframe Connect Client Option uses traditional Open Client APIs.

The Mainframe Connect Client Option allows you to treat the mainframe as if it were just another node on a LAN.

How TRS differs from a DB2 access service

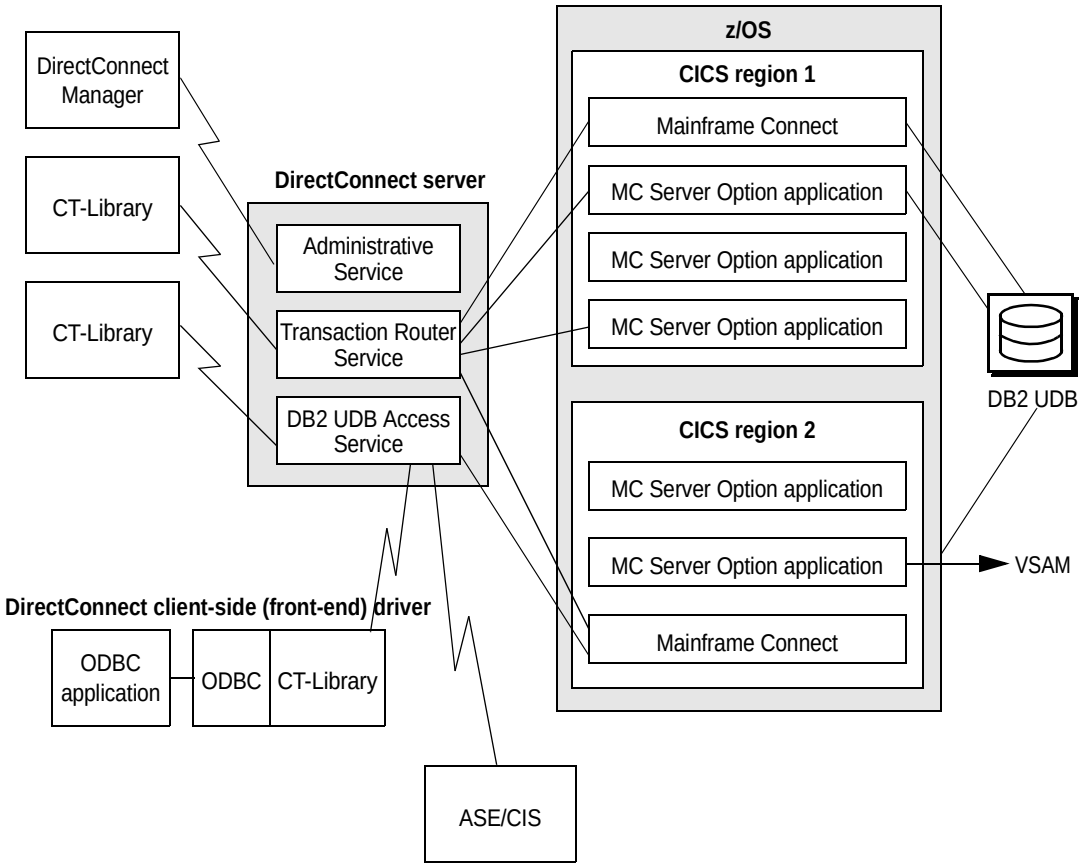
Similar to the DB2 service library component of DirectConnect for z/OS Option, TRS allows users to access DB2 UDB data. They both perform protocol translation, route client requests and server results, and allow remote mainframe password management.

As shown in Figure 1-3, a DB2 access service allows the client application to access data stored in a DB2 UDB database running on z/OS through the Mainframe Connect DB2 UDB Option, a CICS transaction. However, the DB2 access service cannot invoke other CICS transactions. In direct contrast, TRS allows the client access—through Mainframe Connect—to invoke CICS, IMS, and MVS transactions that are based on Mainframe Connect Server Option APIs.

As a result, a single client connection through TRS can access many CICS transactions in multiple CICS regions.

Note Due to space considerations in the diagram, “Mainframe Connect Server Option application” is shortened to “MC Server Option application.”

Figure 1-3: TRS accessing many CICS transactions



In addition, TRS provides:

- Additional security control on a user or transaction basis
- Access to IMS and MVS data
- Access to multiple Mainframe Connect Server Option-based CICS transactions in multiple CICS regions, including any Mainframe Connect running in the defined regions.

A DB2 access service provides:

- Access to remote stored procedure (RSP) programs (TRS does not)
- DB2 and SQL datatype transformation

- Access to bidirectional transfer functionality
- Advanced datatype conversion

Basically, you use TRS when:

- Your applications invoke Mainframe Connect Server Option-based mainframe transactions through remote procedure calls (RPCs) or language requests.
- You use client applications written for the TRS predecessor, Net-Gateway.

DirectConnect Manager

DirectConnect Manager is a graphical user interface (GUI) systems management tool for administering DirectConnect for z/OS Option. It allows you to:

- Manage DirectConnect servers on multiple platforms
- Change configuration properties of DirectConnect servers, service libraries, and access services
- Create and copy services
- Create new servers using DCDirector
- Start and stop existing servers using DCDirector
- Start, stop, and delete services
- 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

CHAPTER 2

Creating a TRS Library

Configuring a TRS library

Note You can use DirectConnect Manager to create and edit a TRS configuration file.

To create and modify a TRS library, you must edit the TRS configuration file. It is a simple text file named *srvlibname.cfg*, where *srvlibname* is the base name of the TRS executable file. For example, if you have the sample LU 6.2-based TRS library, your default configuration file is called *TRSLU62.CFG*. You can use a text editor to change any service library property by editing and saving this file, located as follows:

- For Microsoft Windows:

  `C:\<install_dir>\DC-15_0\servers\srvname\cfg`

- For Linux and UNIX:

  `/<install-dir>/DC-15_0/servers/srvname/cfg`

Running DirectConnect for z/OS Option with the `-N` option as part of the initial configuration will create a sample TRS configuration file that you can modify for your site. For information about installing the sample service library, see the Mainframe Connect DirectConnect for z/OS Option *Installation Guide*.

When you edit the TRS configuration file:

- Enter values for configuration properties that apply to that TRS only. The system ignores properties that are not applicable to your installation. For example, the ConnInfoFile property applies *only* to LU 6.2 installations and not to TCP/IP installations.
- Ignore properties for which the default value is sufficient.
  See “Service library configuration properties” on page 20 for the default values of each configuration property.

The TRS library uses some configuration information from the DirectConnect server. For configuration instructions, see the ECDA and Mainframe Connect *Server Administration Guide*.

Sample TRS configuration file

This is an example of a TRS configuration file:

```
[Service Library]
{Transaction Router Service Property}
PEMDest=CICSQA
RPCInfoFile=C:\<install_dir>\
   DC-15_0\servers\srvname\cfg\trslu62.rpc
LogInfoFile=C:\<install_dir>\
   DC-15_0\servers\srvname\log\trslu62.grp
TDSTraceFile=C:\<install_dir>\
   DC-15_0\servers\srvname\log\trslu62.tds
AccountFile=C:\<install_dir>\
   DC-15_0\servers\srvname\log\trslu62.act
MaxConnections=100
TraceTRS=short
Security=no
DirectPrevent=yes
Accounting=yes
UseDBRPC=no
TruncateLV=no
UpperCase=no
ConnInfoFile=C:\<install_dir>\
   DC-15_0\servers\srvname\cfg\trslu62.cid
ConQTimeout=120
DeactCon=no
 [ServiceA]
{Tranaction Router Service Property}
 EnableAtStartup=yes
```

TRS configuration file format

These principles apply to TRS configuration properties:

- Service library properties apply to the service library as a whole. TRS configuration properties are service library properties, except the service-level properties Description, EnableAtStartup and ClientIdleTimeout. How a TRS service operates is affected by the values of its parent service library.
- Configuration properties are not case sensitive. In this guide, property names appear in mixed case for easier reading.

A TRS configuration file consists of the following lines. (For a sample file, see "Sample TRS configuration file" on page 12.)

- The name of the TRS service library is shown in brackets on the first line of the file. This character string must appear at the top of the file.
- The subsection name, Transaction Router Service Property, is shown in braces on the next line. This character string must appear under the service library line. There are no other subsections.
- Each configuration property and its value are shown on individual lines. Configuration properties can be listed in any order within their subsection. If a configuration property line is deleted or omitted from the file for any reason, the default value for that property is applied automatically. See "Service library configuration properties" on page 20 for the default values of each configuration property.
- The TRS service name is shown in brackets and must conform to the following rules:
  - Service names must be unique within the first 11 characters in length.
  - The initial character must be an alphabetic character (a–z, A–Z).
  - Subsequent characters can be alphabetic or numeric characters or the underscore (_) character.
  - To add a service named "ServiceA," the following line must exist in the TRS configuration file:

```
[ServiceA]
```

For a client to successfully connect to a service, the service name must correspond to a query type entry in the client *interfaces* file. For an explanation of query type entries and how to add them to the interfaces file, see the Mainframe Connect DirectConnect for z/OS Option *Installation Guide*. When a client connects to DirectConnect, it specifies a service name, as shown in the following isql example:

```
isql -Usa -P -SServiceA
go
```

where *ServiceA* is the service name (or server name to the client).

You can include comments in the TRS configuration file. Each comment must be on a separate line and begin with a semicolon or "#" symbol.

Modifying property values

Most configuration properties have default values. Some properties require that you supply values for your site. You can change the properties by using a text editor or by using DirectConnect Manager.

Using DirectConnect Manager

You can configure the TRS library properties and the TRS service properties by using the DirectConnect Manager. To configure the TRS library and service properties, refer to the DirectConnect Manager online Help topic, Managing Configuration | Modifying server configuration properties.

Note When you use DirectConnect Manager to change Accounting, LogTRS, and TraceTRS properties, changes take effect immediately. Changes to all other TRS library configuration properties take effect when you restart the server.

Using a text editor

❖ **To change the TRS service library and services configuration property values**

1 Open the TRS configuration file and change the service library properties as applicable.

2 Open the TRS service file and change the TRS service property values as applicable.

3 Save the file.

4 Stop the server, and then restart it to implement the changes.

Creating additional TRS configurations

This section describes how to create additional TRS libraries and services.

Creating additional TRS libraries

Because many of the server library specific properties affect the major functions of TRS, you may want to create multiple instances of a TRS Library to obtain functionally different configurations. For example, if you want one TRS LU62 Library service that enforces security and one that does not, create two instances of the TRS LU62 Library. This is necessary because the security configuration property operates at the server library level and affects all services in that library. To copy a TRS library, use the trscopy utility, described in the following subsection.

Using the trscopy utility

The trscopy utility creates a copy of a DirectConnect TRS Library by using an existing TRS as a template to find all of the related source files.

Note This utility makes a copy of a TRS service library executable available for all DirectConnect servers defined under the installation area, but it sets up a sample service *only* under the same DirectConnect server as provided on the command line.

The trscopy utility copies files with a base file name of the source TRS in the directory tree of the source DirectConnect server to files with a base file name of the target TRS in the directory tree of the target DirectConnect server.

For example, if you are using Windows, to create a new instance of the TRS LU62 Library (named *new_trslu62*), run the trscopy program with the appropriate arguments. Doing this produces two complete sets of the TRS LU62 files:

- The original *trslu62* executable and all its files

- An executable named *new_trslu62* and a copy of all of its files, with a base file name of *new_trslu62*, placed in the destination DirectConnect server subdirectory tree

Table 2-1 shows the TRS executable file names based on the platform:

Table 2-1: File names for TRS libraries by platform

Platform	File name and extension
Windows	*trslu62.dll and trstcp.dll*
HP	*trslu62.sl and trstcp.sl*
AIX	*trslu62.so and trstcp.so*
Solaris	*trslu62.so and trstcp.so*

After you run trscopy, you must edit the new TRS Library's configuration file and change the service name. Then, add this new service name to the SYBASE interfaces file on the client. For instructions, see the Mainframe Connect DirectConnect for z/OS Option *Installation Guide*.

Note The UNIX version of trscopy has an option that automatically generates a new TRS name in the target TRS library configuration file. Then, it adds that service to the SYBASE *interfaces* file on the server.

Using the trscopy command

This is the trscopy command for Microsoft Windows:

```
trscopy source_DirectConnect source_Service_Library
destination_DirectConnect destination_Service_Library
```

This is the trscopy command for UNIX:

```
trscopy.sh source_DirectConnect source_Service_Library
destination_DirectConnect destination_Service_Library
[-a] [-l] [-v]
```

Table 2-2 describes the parameters and options in the trscopy command.

Table 2-2: Description of trscopy parameters and options

Parameter	Description
source_DirectConnect	(*Required*) Name of the server that contains the source TRS files. It is located in the directory under the following name: For Windows: • *C:\<install_dir>\DC-15_0\servers* • For UNIX: /*<install_dir>/DC-15_0/servers*

Parameter	Description
source_TRS_Library	(*Required*) The name of the source TRS library associated with this server. It must exist under the source DirectConnect server name.
destination_DirectConnect	(*Required*) Name of the target DirectConnect server, located in the directory under *C:\<install_dir>\DC-15_0\servers.*
destination_TRS_Library	(*Required*) Name of the target TRS library. Follow these guidelines: • The executable associated with the server library name must not exist under the target DirectConnect server. • The file name must not exist anywhere under the following *locales* directory structure, based on platform: For Windows: *C:\<install_dir>\DC-15_0\connectivity\locales* For UNIX: */<install_dir>/DC-15_0/connectivity\locales*
-a	(*Optional and for UNIX only*) Option that generates new service names for the new service library and adds these new services to the *interfaces* file.
-l	(*Optional and for UNIX only*) A "softlink" option. When possible, use softlinks instead of copying a file.
-v	(*Optional and for UNIX only*) Verbose option, which displays a description of each operation while it runs.

Warning! If you used trscopy to create custom-named service libraries in */<install_dir>/DC-15_0/svclib*, when you perform an upgrade, only the standard-named service libraries are upgraded. To correct this situation, do one of the following:

- Run trscopy again, or
- Copy the standard library file to the custom library file.

Example

This UNIX example shows the original image on the hard drive, followed by using trscopy, and the resulting image on the hard drive.

```
belford> cd /<install_dir>/DC-15_0

belford> source ./DC_SYBASE.csh
belford> cd /<install_dir>/DC-15_0/svclib

belford> ls
admin.so trs.sif trslu62.so trstcp.so

belford> cd /<install_dir>/DC-15_0/bin

belford> cat $SYBASE/interfaces
d3s1261e2bf
master tcp ether belford 12612
```

```
query tcp ether belford 12612

belford> ./trscopy.sh srvname trstcp srvname
    trstcp02 -a -v /<install_dir>/DC-
15_0/svclib/trstcp02.*: No such file or directory
*******************************************************

To complete the service library copy process, change the
service names in /install_dir>/DC-
15_0/servers/srvname/cfg/trstcp02.cfg and add these new
services to /<install_dir>/DC-
15_0/connectivity/interfaces.

*******************************************************

./trscopy.sh completed successfully.

*******************************************************

belford> cat /<install_dir>/DC-
15_0/servers/srvname/cfg/trstcp02.cfg

[Service Library]

{Transaction Router Service Property}

TraceProcessUserExits=no
ProcessExitFile=
ProcessExitEnabled=no
UserExitFile=/<install_dir>/DC-
15_0/servers/srvname/log/ngue.trstcp
UserExit=no
UseDBRPC=yes
UpperCase=yes
UpgradePassword=no
TruncateLV=no
TraceProtocol=none
TraceTRS=none
TDSTraceFile=/<install_dir>/DC-
15_0/servers/srvname/log/ngtds.trstcp
Send5701=no
Security=no
RPCInfoFile=/<install_dir>/DC-
15_0/servers/srvname/cfg/ngrpc.trstcp
RegionInfoFile=/<install_dir>/DC-
15_0/servers/srvname/cfg/ngreg.trstcp
ProtocolTraceFile=/<install_dir>/DC-
15_0/servers/srvname/cfg/ngtcp.trstcp
PEMDestType=CICS
PEMDest=
```

```
MaxConnections=25
LogTRS=yes
LogInfoFile=/<install_dir>/DC-
15_0/servers/srvname/cfg/nggrp.trstcp
DirectPrevent=no
Description=Sybase Transaction Router Service Library
DeactCon=yes
ConQTimeout=60
ConnInfoFile=/<install_dir>/DC-
15_0/servers/srvname/cfg/ngcid.trstcp
Accounting=no
AccountFile=/<install_dir>/DC-
15_0/servers/srvname/cfg/ngact.trstcp
[tt12538ss]

{Transaction Router Service Property}
EnableAtStartup=yes

belford> cd /home/<install_dir>/DC-15_0/svclib
belford> ls
admin.so trs.sif trslu62.so trstcp.so trstcp02.50
```

Creating additional TRS services

You can create additional services using DirectConnect Manager or by using your text editor.

To create a new service using DirectConnect Manager, go to DirectConnect Manager Online Help | Managing Access Services | Creating a New Service, or follow the instructions in this section.

Using a text editor

❖ **To create and change services using a text editor**

1 Open the TRS configuration file from one of the following directories:

- For Windows:

```
C:\<install_dir>\DC-15_0\servers\srvname\trs_service_lib.cfg
```

- For UNIX:

```
/<install_dir>/DC-15_0/servers/srvname/trs_service_lib.cfg
```

2 Add the service name in brackets below the initial service names section.

3 By default, services are not enabled for client connection at start-up. If you want this service to be enabled at start-up, add the EnableAtStartup configuration property, set to yes, below the new service name.

4 Save the file.

5 Stop the server, and then restart it to implement the changes.

6 To be sure that client applications can connect to a new TRS service from a client machine, you must enter the service name in the SYBASE *interfaces* file on the client machine. If you choose to use the Service Name Redirection utility, make an assigned service name entry in the *Service Name Redirection* file.

For information about service name redirection, see the Enterprise Connect Data Access and Mainframe Connect *Server Administration Guide*.

Using DirectConnect Manager

❖ **To create a new service using DirectConnect Manager**

1 Start DirectConnect Manager.

2 Double-click the server name.

3 Right-click the Services folder and select Create Service.

4 Select TRS-LU62 or TRS-TCP/IP from the Type of Services window and enter the name of your new TRS service.

5 Click Finish. The new service name is displayed

Service library configuration properties

Table 2-3 lists all TRS service library configuration properties and identifies the location for a more detailed description in this chapter:

Table 2-3: TRS configuration properties

TRS configuration property	TRS configuration property description	Location
AccountFile	Specifies the directory, path, and file name to which accounting records are written.	“AccountFile” on page 22
Accounting	Turns accounting on and off.	“Accounting” on page 23

TRS configuration property	TRS configuration property description	Location
ConnInfoFile (LU 6.2 only)	Specifies the directory, path, and file name that contains LU 6.2 connection information for this TRS.	"ConnInfoFile" on page 23
ConQTimeout (LU 6.2 only)	Specifies the LU 6.2 connection queue timeout period (wait period) in seconds.	"ConQTimeout (LU 6.2 only)" on page 24
DeactCon (LU 6.2 only)	Indicates whether an LU 6.2 connection should be deactivated or left active if a line failure or other error occurs.	"DeactCon (LU 6.2 only)" on page 24
Description	An optional customer-supplied description of the service library.	"Description" on page 25
DirectPrevent	Instructs TRS to accept requests from Adaptive Server clients only.	"DirectPrevent" on page 25
LogInfoFile	Specifies the directory, path, and file name that contains client login and security group information.	"LogInfoFile" on page 25
LogTRS	Enables or disables logging to the server log file.	"LogTRS" on page 26
MaxConnections	Specifies the maximum number of clients that can be logged into this TRS library concurrently.	"MaxConnections" on page 26
PEMDest	Specifies the destination region handling the IBM Password Expiration Management (PEM) transaction program. Applies to LU 6.2 only.	"PEMDest" on page 27
PEMDestType	Specifies the type of destination region (*PEMDest*) managing the PEM server transaction. Applies to LU 6.2 only.	"PEMDestType" on page 27
ProcessExitEnabled	Enables the use of process user exits.	"ProcessExitEnabled" on page 27
ProcessExitFile	Identifies the path and name of the shared library that you have created.	"ProcessExitFile" on page 28
RegionInfoFile (TCP/IP only)	Specifies the directory path and file name for the file that contains TCP/IP connection information for this TRS.	"RegionInfoFile" on page 29
ReturnParametersOnError	Specifies whether to return parameters when an error occurs.	"ReturnParametersOnError" on page 29
RPCInfoFile	Specifies the directory, path, and file name of the file containing the remote procedure call (RPC) information for this TRS.	"RPCInfoFile" on page 29
Security	Tells TRS whether to validate logins against its own login information in addition to the validation done by the mainframe.	"Security" on page 30
Send5701	Indicates whether the message 5701 should be sent back to the client for use database statements.	"Send5701" on page 30

TRS configuration property	TRS configuration property description	Location
TDSTraceFile	Specifies the directory, path, and name of the file to which TDS information is written.	"TDSTraceFile" on page 31
TraceProcessUserExits	Traces entry/exit points of function call to each process user exit you have created.	"TraceProcessUserExits" on page 31
TraceTRS	Specifies the level of TDS tracing that TRS is to record.	"TraceTRS" on page 32
TruncateLV	Truncates any mainframe long varchar fields to 255 bytes before sending them to the client.	"TruncateLV" on page 32
UpgradePassword	Indicates whether pre-TRS passwords (8 bytes maximum) should be upgraded to the new format (30 bytes maximum).	"UpgradePassword" on page 33
UpperCase	Automatically changes lowercase user IDs and passwords to uppercase for users logged into the LAN before forwarding these values to the mainframe.	"UpperCase" on page 33
UseDBRPC	Allows a client to send RPC requests larger than 64K to the mainframe.	"UseDBRPC" on page 34

The following subsections, in alphabetical order, describe each TRS configuration property in Table 2-2.

AccountFile

Specifies the directory, path, and name of the file where TRS stores accounting records written to it. See "Controlling accounting" in Chapter 7, "Controlling a TRS," for more information about the type of accounting information that TRS captures.

Syntax

AccountFile=*newpath*

where *newpath* is the directory, path, and name of the file to which TRS writes accounting records.

Default

- For an LU 6.2-based TRS on Windows, *newpath* is:

```
C:\install_dir\DC-15_0\servers\srvname\log\srvlibname.act
```

- For a TCP/IP-based TRS on Windows, *newpath* is:

```
C:\install_dir\DC-15_0\servers\srvname\log\srvlibname.act
```

- For a TRS on UNIX platforms, *newpath* is:

/<install_dir/DC-15_0/servers/*srvname*/log/ngact.*srvlibname*

Values

srvname is the name of the DirectConnect server defined during server installation.

srvlibname is the name of the TRS library.

Accounting

Turns accounting on or off. TRS writes accounting records to the file specified in the AccountFile configuration property.

TRS users with administrative privileges can also turn accounting on and off using the sgw_startact and sgw_stopact procedures while TRS is running.

Note Using DirectConnect Manager, you can turn accounting on and off either at server start-up or dynamically while TRS is running.

Syntax

Accounting=[no | yes]

Default

no

Values

no turns accounting off.

yes turns accounting on.

ConnInfoFile

Specifies the directory, path, and name of the file that contains LU 6.2 connection information for this TRS. This file is created the first time you define an LU 6.2 connection. A default is defined for TCP/IP on non-UNIX platforms. (To define connections, use the sgw_addcon procedure.)

Syntax

ConnInfoFile=*newpath*

where *newpath* is the directory path and name of the TRS LU 6.2 connection information file.

Default

- For an LU 6.2-based TRS on Windows, *newpath* is:

C:\<install_dir>\DC-15_0\servers*srvname*\cfg*srvlibname*.cid

- For a TCP/IP-based TRS on Windows, *newpath* is:

C:\<install_dir>\DC-15_0\servers*srvname*\log*srvlibname*.ngcid

- For a TRS on UNIX platforms, *newpath* is:

```
/<install_dir>/DC-15_0/servers/srvname/cfg/ngcid.srvlibname
```

Values

srvname is the name of the DirectConnect server defined during server installation.

srvlibname is the name of the TRS library.

ConQTimeout (LU 6.2 only)

Specifies the number of seconds client requests are allowed to wait in a queue for an available LU 6.2 connection to the destination system. A client request times-out (expires) if a connection does not become available in the specified length of time.

Syntax

ConQTimeout=*timeout*

where *timeout* is the maximum number of seconds that each client request remains in a queue to wait for an available LU 6.2 connection to the destination system before the client request expires.

Range

0 to 50000

Default

60

Comments

Specify 0 (zero) if you do not want client requests to queue up. When a timeout occurs, TRS returns a message to the client.

DeactCon (LU 6.2 only)

Indicates whether TRS deactivates or leaves active an LU 6.2 connection if a line failure or other error occurs on that connection.

If you specify deactivation, TRS marks the failing connections as inactive when an LU 6.2 error occurs. You can restart connections while TRS is running by using the sgw_actcon procedure.

Syntax

DeactCon=[no | yes]

Default

yes

Values

yes deactivates a connection when an LU 6.2 error occurs on the connection.

no leaves a connection active even after an LU 6.2 error occurs on the connection.

Description

Provides an optional customer-supplied description of the service library.

Syntax

Description=*description*

description where is the description of the service library up to 255 alphanumeric characters.

Default

A blank string, for example:

```
Description=
```

Comments

Specifying the default sets the value to a blank string.

DirectPrevent

Instructs TRS to reject all direct requests from a client, forcing clients to route all requests through Adaptive Server. Setting this property to no allows clients to send RPCs and language requests directly to TRS.

Syntax

DirectPrevent=[no | yes]

Default

no

Values

yes rejects all direct requests from the client and forces the client to route all requests through Adaptive Server.

no allows clients to send RPCs and language requests directly to TRS.

LogInfoFile

Specifies the directory, path, and name of the file that contains client login and security group information. TRS creates this file the first time you define a client login, connectivity group, or transaction group. (To define client logins, use the sgw_addlog procedure.)

Syntax

LogInfoFile=*newpath*

where *newpath* is the directory path and file name of the file containing the client login and security group information for this TRS library.

Default

- For an LU 6.2-based TRS on Windows, *newpath* is:

```
C:\<install_dir>\DC-15_0\servers\srvname\cfg\srvlibname.grp
```

- For a TCP/IP-based TRS on Windows, *newpath* is:

C:\<install_dir>\DC-15_0\servers*srvname*\cfg*srvlibname*.nggrp

- For UNIX platforms, *newpath* is:

/<install_dir>/DC-15_0/servers/*srvname*/cfg/nggrp.*srvlibname*

Values

srvname is the name of the DirectConnect server defined during server installation.

srvlibname is the name of the TRS library.

LogTRS

Enables or disables logging to the server log file, located at:

- For UNIX:

/<install_dir>/DC-15_0/servers*srvname*/log/*srvlibname*.log

- For Windows:

C:\<install_dir>\DC-15_0\servers*srvname*\log*srvlibname*.log

Syntax

LogTRS=[no | yes]

Default

no

Values

yes turns logging on.

no turns logging off.

MaxConnections

Specifies the maximum number of clients that can be logged into this TRS library concurrently.

Syntax

MaxConnections=*integer*

where *integer* is a number of clients.

Range

1–*n*, where *n* is the maximum number of clients allowed by the server. (For more information, see “MaxConnections” in the DirectConnect *Server Administration Guide*.)

Default

25

Comments

TRS does not verify the validity of this number.

PEMDest

Specifies the destination system for the IBM Password Expiration Management (PEM), a password management program that IBM provides.

Sybase provides support for PEM as a feature of TRS for LU 6.2. This feature is not available for TRS connections to the mainframe using TCP/IP.

For more information about implementing PEM, see Chapter 6, "Using Password Expiration Management (PEM) with TRS."

Syntax PEMDest=*destsys*

where *destsys* is system-dependent value identifying the LU 6.2 connection from which the IBM PEM sign-on transaction can be accessed. Use the value supplied for the *region* parameter when this LU 6.2 connection was defined with the sgw_addcon procedure.

Default No default. No value is required unless clients are using TRS PEM support.

Comments Leaving the default blank string in place disables PEM RPCs.

PEMDestType

Specifies the type of destination region managing the PEM server transaction as defined by the PEMDest configuration property.

Syntax PEMDestType=[CICS | MVS]

Default CICS

Values CICS indicates that the PEMDest value connects to a CICS region.

MVS indicates that the PEMDest value connects to native MVS.

ProcessExitEnabled

Enables the use of process *user exits*. Only the exits that you have defined and added to your *user exit* library will be invoked.

Syntax ProcessExitEnabled=[yes | no]

Default *no*

Values *yes* enables the use of process user exits.

no does not allow process user exits to be invoked.

ProcessExitFile

Provides the full path and name of the *user exit* shared library that you have created.

Syntax

ProcessExitFile=[*path* / *filename* | null]

Default

null

Values

path / filename identifies the full path and file name of the *user exit* shared library that you created.

null indicates that no process *user exit* shared library has been created.

ProtocolTraceFile

Specifies the directory path and file name in which DirectConnect uses the back-end transport protocol traces and errors for conversations between the TRS library and the mainframe.

Syntax

TDSTraceFile=*newpath*

where *newpath* is the directory path and file name to which traces are written for TRS.

Default

- On Windows, back-end TCP/IP tracing goes into the *newpath*:

```
C:\<install_dir>\DC-15_0\servers\srvname\log\trstcp.ngtcp
```

- On UNIX, back-end TCP/IP tracing goes into the *newpath*:

```
/<install_dir>/DC-15_0/servers/srvname/log/ngtcp.trstcp
```

- On Windows, back-end LU 6.2 tracing goes into the *newpath*:

```
C:\<install_dir>\DC-15_0\servers\srvname\log\trslu62.nglu62
```

- On UNIX, back-end LU6.2 tracing goes into the *newpath*:

```
/<install_dir>/DC-15_0/servers/srvname/log/nglu62.trslu62
```

Values

srvname is the name of the DirectConnect server defined during server installation.

RegionInfoFile

Specifies the directory, path, and name of the file containing the TCP/IP connection information that it creates the first time you define a TCP/IP region. A default has been defined for LU 6.2-based TRS on Windows. (To define regions, use the sgw_addregion procedure.)

Syntax

RegionInfoFile=*newpath*

where *newpath* is the directory path and file name of the file containing the TRS TCP/IP connection information.

Default

- For an LU 6.2-based TRS on Windows, *newpath* is:

`C:\<install_dir>\DC-15_0\servers\srvname\cfg\srvlibname.reg`

- For a TCP/IP-based TRS on Windows, *newpath* is:

`C:\<install_dir>\DC-15_0\servers\srvname\cfg\srvlibname.ngreg`

- For UNIX platforms, *newpath* is:

`/<install_dir>/DC-15_0/servers/srvname/cfg/ngreg.srvlibname`

Values

srvname is the name of the DirectConnect server defined during server installation.

srvlibname is the name of the TRS library.

ReturnParametersOnError

Provides the ability to request that parameters be returned when an error occurs.

Syntax

ReturnParametersOnError=[no | yes]

Default

yes

Values

- yes returns parameters when an error occurs.
- no returns errors with no parameters.

RPCInfoFile

Specifies the directory path and file name of the file containing the RPC information that TRS creates the first time you define an RPC. (To define RPCs, use the sgw_addrpc procedure.)

Syntax

RPCInfoFile=*newpath*

where *newpath* is the directory path and file name of the file containing the RPC information for this TRS.

Default

- For an LU 6.2-based TRS on Windows, *newpath* is:

```
C:\<install_dir>\DC-15_0\servers\srvname\cfg\srvlibname.rpc
```

- For a TCP/IP-based TRS on Windows, *newpath* is:

```
C:\<install_dir>\DC-15_0\servers\srvname\cfg\srvlibname.ngrpc
```

- For UNIX platforms, *newpath* is:

```
/<install_dir>/DC-15_0/servers/srvname/cfg/ngrpc.srvlibname
```

Values

srvname is the name of the DirectConnect server defined during server installation.

srvlibname is the name of the TRS library.

Security

Tells TRS whether to validate logins against its own login information in addition to the validation done by the mainframe. If you set the Security configuration property to no, TRS transparently forwards the user ID and password to the mainframe (based on the RPC definition security parameter values defined in the sgw_addrpc procedure). You do not need to use the sgw_addlogsecurity procedure to add new users to TRS when you set Security to no.

Syntax

Security=[no | yes]

Default

yes

Values

yes turns security on.

no turns security off.

Send5701

Indicates whether message 5701 should be sent to the client for use database statements.

Syntax

Send5701=[no | yes]

Default

no

Values

yes sends message 5701 to the client for use database statements.

no does not send message 5701 to the client for use database statements.

TDSTraceFile

Specifies the directory path and file name in which the Tabular Data Stream™ (TDS) records traces and errors for conversations between the TRS library and the mainframe. TDS is the Sybase application-level protocol that defines the form and content of relational database requests and replies.

Syntax

TDSTraceFile=*newpath*

where *newpath* is the directory path and file name to which TDS traces are written for TRS.

Default

- For an LU 6.2-based TRS on Windows, *newpath* is:

`C:\<install_dir>\DC-15_0\servers\srvname\log\srvlibname.tds`

- For a TCP/IP-based TRS on Windows, *newpath* is:

`C:\<install_dir>\DC-15_0\servers\srvname\log\srvlibname.ngtds`

- For UNIX, *newpath* is:

`/<install_dir>/DC-15_0/servers/srvname/log/ngtds.srvlibname`

Values

srvname is the name of the DirectConnect server defined during server installation.

srvlibname is the name of the TRS library.

TraceProcessUserExits

Traces the entry/exit points of the function call to each of the process *user exits* that you have created. Normal setting is no; however, a setting of yes will help you execute your processing user exits.

Syntax

TraceProcessUserExits=[yes | no]

Default

no

Values

yes turns on tracing for the process user exits you have created.

no does not turn on tracing for the process user exits that you have created.

TraceProtocol

Provides protocol level tracing for DirectConnect using LU 6.2 or TCP/IP, depending on which service library is being traced.

Syntax: TraceProtocol=[none | short | long]

Default: none

Values: short or long protocol level tracing is turned on for LU 6.2 or TCP/IP.

none protocol level tracing is not turned on for LU 6.2 or TCP/IP.

TraceTRS

Turns TDS tracing on or off. Using the long or short option turns on tracing. You can also turn tracing on and off when TRS is running using the sgw_starttrace and sgw_stoptrace procedures, with the TDS parameters, or by using DirectConnect Manager. Formatted TDS traces are written to a file defined by the TDSTraceFile configuration property. See “Starting tracing” on page 133.

Note Using DirectConnect Manager, you can turn tracing on and off dynamically while TRS is running or at server start-up.

Syntax: TraceTRS=[none | short | long]

Default: none

Values: long traces both TDS header information and TDS data packets.

short traces TDS header information only.

none turns tracing off.

TruncateLV

Truncates any mainframe long varchar fields to 255 bytes before sending them to the client. Setting this property to no causes long varchar data to be sent as text and image datatypes for 4.x TDS clients, or as the appropriate long varchar datatype for 5.0 TDS clients. (Sybase System 10 and later versions use TDS 5.0.)

Syntax: TruncateLV=[no | yes]

Default no

Values no turns long varchar truncation off, causing long varchar data to be sent as text and image datatypes for 4.x TDS clients, or as the appropriate long varchar datatype for 5.0 TDS clients.

yes turns long varchar truncation on, truncating any mainframe long varchar fields to 255 bytes before sending them to the client.

UpgradePassword

Indicates whether pre-Net-Gateway version 3.0.1 passwords (8 bytes maximum) should be upgraded to the new format (30 bytes maximum). If this property is set to yes, all existing old passwords are lost and are initialized to null.

Syntax UpgradePassword=[no | yes]

Default yes

Values no prevents the upgrading of pre-TRS passwords.

yes upgrades all pre-TRS passwords to the new format, and deletes all existing passwords and initializes them to null.

Comments

- If you are upgrading from Net-Gateway version 2.0, set this configuration property to yes.
- Before setting UpgradePassword to yes and triggering the upgrade, copy your log information files to a *save* directory:

```
cd C:\%<install_dir>%
mkdir save_nggrp
copy log_info_files save_nggrp
```

  For the name of the log information file for your system configuration, see the LogInfoFile configuration property.

UpperCase

Automatically changes lowercase user IDs and passwords to uppercase for clients before forwarding these values to the mainframe.

Syntax UpperCase=[no | yes]

Default yes

Values

no prevents the automatic changing of lowercase user IDs and passwords to uppercase before forwarding these values to the mainframe. TRS forwards the user ID and password as is.

yes automatically changes lowercase user IDs and passwords to uppercase.

UseDBRPC

Allows a client to send RPC requests larger than 64KB to the mainframe.

When an RPC is executed, a client sends a TDS_RPC TDS token stream to the server. The server reads the stream, processes the request and returns any results back to the client. The TDS_RPC stream in TDS version 5.0 had a 2-byte integer indicating the total length of the TDS_RPC stream, which limits each TDS_RPC stream to 64KB in length.

DirectConnect provides a new token, named TDS_DBRPC. This token removes the RPC length limit.

Note If you want to send RPC requests larger than 64K to the mainframe, and you are running Mainframe Connect Server Option (Open ServerConnect) software that pre-dates Open ServerConnect 3.1, check your latest release bulletins to verify that the UseDBRPC property is compatible with your version. If your Mainframe Connect Server Option version is incompatible with the UseDBRPC property, your RPC requests will fail when this feature is on.

Syntax

UseDBRPC=[no | yes]

Default

yes

Values

no turns off DBRPC, preventing a client from sending RPC requests larger than 64K to the mainframe.

yes turns on DBRPC, allowing a client to send RPC requests larger than 64K to the mainframe.

Service configuration properties

Table 2-4 lists the TRS service configuration properties, which are described in following subsections.

Table 2-4: TRS Service configuration properties

TRS service configuration property	TRS service configuration property description	Location
ClientIdleTimeout	Specifies the number of minutes a connected TRS client can remain idle before being disconnected.	“ClientIdleTimeout” on page 35
Description	An optional customer supplied definition of the TRS service.	“Description” on page 36
EnableAtStartup	Specifies whether the TRS service starts and accepts client connections when the DirectConnect server starts.	“EnableAtStartup” on page 36
XNLChar	Specifies the maximum size of both char and binary results.	“XNLChar” on page 36
XNLVarChar	Specifies the maximum size of both varchar and varbinary results.	“XNLVarChar” on page 37

ClientIdleTimeout

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

Syntax — ClientIdleTimeout=*integer*

Range — 0–1024

Default — 0

Values — *integer* is how many minutes a client connection can remain inactive before an access service terminates the connection.

0 indicates that an access service never terminates an idle connection.

Comments

- A connection is idle when:
  - A client is 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 screen paused for the specified timeout period.

- The TRS access service checks client activity one time each minute. Therefore, a client can remain inactive for up to one minute beyond the ClientIdleTimeout value before the TRS access service terminates the connection.

Description

Provides an optional customer-supplied description of the TRS service.

Syntax

Description=*where*

description is the description of the TRS service up to 255 alphanumeric characters.

Default

No default.

Comments

Specifying no value sets the value to a blank string.

EnableAtStartup

Specifies whether this TRS service starts and accepts client connections when the DirectConnect server starts.

Syntax

EnableAtStartup=[no | yes]

Default

yes

Values

no means that the TRS service does not start when the server starts.

yes means that the TRS service starts when the server starts.

Comments

If you are not using DirectConnect Manager to manage your access services, set this property to yes.

XNLChar

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

Syntax

XNLChar=*integer*

Default	256
Values	*integer* is a valid number between 256 - 2147483647 (two gigabytes).
Comments	Sybase recommends that this value match the maximum size of the char and binary datatypes of the back end database. It is common for this limit to be the same for the char and binary datatypes.

XNLVarChar

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

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

CHAPTER 3

Configuring a TRS

Using TRS administration procedures

TRS administration procedures begin with sgw_, which stands for server gateway.

The TRS administration procedures are isql execute commands. Start isql or your preferred dynamic SQL utility as usual, and then enter the commands at the prompt.

Command conventions

Observe the following conventions when you use TRS administration procedures:

- Run each exec command individually; do not batch them.
- Enter go after each command, or execute the command according to the conventions of your SQL utility. (Generally, go is not shown in the syntax illustrations in this guide; it is shown in the examples.)
- Enclose command parameters that contain numerical values in quotation marks.

- Enter all command parameters in the order shown. Separate the parameters with commas. (Spaces are optional.)

  If you omit any parameters, include the commas as placeholders or use the keyword NULL (not case sensitive). For example, to assign the TRS user "WAYNE" a password of "BLEUCHEZ":

  ```
  exec sgw_chpwd  WAYNE, , BLEUCHEZ
  go
  ```

  or

  ```
  exec sgw_chpwd WAYNE, NULL, BLEUCHEZ
  go
  ```

  The first parameter after sgw_chpwd is the login, in this case, "WAYNE." The second comma (or null) holds the place of the TRS password, which you are not changing. The parameter called "BLEUCHEZ" represents the new password that is passed to the mainframe.

- When entering TRS administration procedures, you need to enter only as many characters as required to make each parameter distinct from any other (you must enter at least three characters). For example, use one of the following to query the status of the clients on TRS:

  ```
  execute sgw_status clients
  ```

  or

  ```
  exec sgw_status cli
  ```

Viewing command results

The results of the administration procedures display on the screen where you entered the command. If the results take up more lines than one screen can display, the information may scroll by quickly (depending on your SQL utility). In this case, you can use your operating system utilities to direct the results of the procedure to a file.

Quick reference to TRS administration procedures

This section provides a quick reference to the administration procedures available for TRS. Sorted by type of procedure, the tables list the object to be operated on, the procedure to use, and a location you can access for detailed information.

In these procedure tables, the parameter values you should replace with the appropriate values for your site are shown in *italics*. Parameters shown in uppercase should be entered in UPPERCASE.

Help procedure

To display an online listing of the command syntax for TRS administration procedures, use the exec command, as shown in the following isql example:

```
exec sgw_help
go
```

The results show a list of the commands, with a short description and syntax for each, identifying all optional entries.

Procedure tables

You may find it useful to photocopy the following tables and post them near your workstation for easy reference. The tables provided on the following pages are listed here:

- For add/drop procedures for client, connection, login, region, RPC, and transaction group, refer to Table 3-1 on page 42.
- For change procedures for login (password) and transaction group, refer to Table 3-2 on page 43.
- For display and status procedures for accounting, client, connection, login, region, RPC, parameter, trace and transaction group, refer to Table 3-3 on page 43.
- For start and stop procedures for accounting, connection, TRS, region, RPC, and trace, refer to Table 3-4 on page 44.
- For Password Expiration Management (PEM) procedures for login (host password) and group login (host password) refer to Table 3-5.

Add/drop procedures

Table 3-1 shows add/drop procedures for client, connection, login, region, and RPC:

Table 3-1: Add/drop procedures

Element	Procedure	Location
Client	sgw_disclient "*client_number*"	"Disconnecting a client" on page 131
Connection (LU 6.2 only)	sgw_addcon *con_name*, *region*, *mode*, "*max_sessions*"	"Adding a connection configuration" on page 48
	sgw_dropcon *con_name*	"Dropping a connection configuration" on page 49
	sgw_dropcon *con_name*, *region*	"Dropping individual regions from a connection configuration" on page 50
	sgw_addcongrp *group_name*	"Adding a connection group" on page 110
	sgw_dropcongrp *group_name*	"Dropping a connection group" on page 111
	sgw_addcontogrp *group_name*, *con_name*	"Adding connections to a connection group" on page 110
	sgw_dropconfromgrp *group_name*, *con_name*	"Dropping connections from a connection group" on page 111
Login	sgw_addlog *login*, *pwd*, *HOST_LOGIN*, *HOST_PWD*, *tran_group*, *con_group*, *gwctrl* **Note** *con_group* is for LU 6.2 only. For TCP/IP, include a comma or null as a placeholder.	"Adding a login" on page 105
	sgw_droplog *login*	"Deleting a user definition" on page 107
Region (TCP/IP only)	sgw_addregion *region*, *HOSTNAME*, *"port_number"*	"Defining regions to TRS" on page 50
	sgw_addregion *region*, *HOSTNAME*, *"port_number"*, *regiontype* (CICS, IMS, MVS)	"Defining regions to TRS" on page 50
	sgw_dropregion *region*	"Dropping a region" on page 51

Element	Procedure	Location
RPC	sgw_addrpc *rpc_name*, *TRAN_ID*, *region*, *security* (none|userid|both)	"Adding an RPC" on page 52
	sgw_droprpc *rpc_name*	"Dropping an RPC" on page 54
	sgw_addrpctogrp *tran_group*, *rpc_name*, *rpcpwdlevel* (none|user|group)	"Adding RPCs to a transaction group" on page 116
	sgw_droprpcfromgrp *tran_group*, *rpc_name*	"Deleting RPC names from a transaction group" on page 117
Transaction group	sgw_addtrngrp *tran_group*, *GROUP_LOGIN*, *GROUP_PWD*, *langrpc*, *langpwdlevel* (none|user|group)	"Adding a transaction group" on page 114
	sgw_droptrngrp *tran_group*	"Deleting a transaction group" on page 118

Change procedures

Table 3-2 shows change procedures for login (password) and transaction group:

Table 3-2: Change procedures

Element	Procedure	Location
Login (password)	sgw_chpwd *login*, *pwd*, *HOST_PWD*	"Changing passwords" on page 106
Transaction group	sgw_modtrngrp *tran_group*, *GROUP_LOGIN*, *GROUP_PWD*, *langrpc*, *langpwdlevel* (none|user|group)	"Modifying a transaction group" on page 117

Display and status procedures

Table 3-3 shows display and status procedures for accounting, client, connection, login, region, RPC, parameter, and trace and transaction group:

Table 3-3: Display/status procedures

Element	Procedure	Location
Accounting	sgw_dspact	"Reading the accounting log" on page 135
Client	sgw_status clients	"Monitoring clients" on page 138
	sgw_status summary	"Summary of clients in each listed state" on page 144

Element	Procedure	Location
Connection (LU 6.2 only)	sgw_status connections	"Monitoring connections (LU 6.2 only)" on page 139
	sgw_dspcongrp	"Displaying one connection group" on page 110
	sgw_dspcongrp *con_group*	"Displaying one connection group" on page 110
Login	sgw_dsplog	"Displaying current logins" on page 104
Region (TCP/IP only)	sgw_status region	"Monitoring regions (TCP/IP only)" on page 140
RPC	sgw_status rpc	"Monitoring RPCs" on page 141
Parameter	sgw_status parameters	"Displaying TRS configuration properties" on page 142
Trace	sgw_status trace	"Requesting trace information" on page 144
Transaction group	sgw_dsptrngrp	"Displaying all transaction groups" on page 113
	sgw_dsptrngrp *tran_group*	"Assigning transaction groups" on page 111
	sgw_dsptrngrp *tran_group*, rpc	"Assigning transaction groups" on page 111

Start and stop procedures

Table 3-4 shows start and stop procedures for accounting, connection, TRS, region, RPC, and trace:

Table 3-4: Start/stop procedures

Element	Procedure	Location
Accounting	sgw_startact	"Activating and deactivating accounting" on page 135
	sgw_stopact	"Activating and deactivating accounting" on page 135
Connection (LU 6.2 only)	sgw_actcon all	"Restarting all connections" on page 128
	sgw_actcon "*con_number*"	"Activating a single connection" on page 128
	sgw_deactcon "*con_number*"	"Deactivating a connection" on page 128
	sgw_deactcon "*con_number*", force	"Deactivating a connection" on page 128

Element	Procedure	Location
TRS	sgw_shutdown	“Stopping TRS” on page 135
	sgw_shutdown now	“Stopping TRS” on page 135
Region (TCP/IP only)	sgw_actregion *region*	“Activating a single region” on page 130
	sgw_actregion all	“Activating a region” on page 130
	sgw_deactregion *region*	“Deactivating a region” on page 130
RPC	sgw_actrpc *rpc_name*	“Activating an RPC” on page 131
	sgw_deactrpc *rpc_name*	“Deactivating an RPC” on page 132
Trace	sgw_starttrace *PROT*	“Starting tracing” on page 133
	sgw_starttrace *TDS*	
	sgw_stoptrace *PROT*	“Stopping tracing” on page 133
	sgw_stoptrace *TDS*	

Password Expiration Manager (PEM) procedures

Table 3-5 shows Password Expiration Management (PEM) procedures for login (host password) and group login (host password):

Table 3-5: PEM procedures

Element	Procedure	Location
Login (Host password)	sqw_peminfopwd *host_login*, *host_password*	“Obtaining information about passwords” on page 121
Login (Host password)	sgw_pemchpwd *new_password*, *new_password*	“Changing an individual password” on page 123
Group login (Host password)	sgw_peminfogrppwd *tran_grp*	“Group password” on page 122
Group login (Host password)	sgw_pemchgrppwd *tran_grp*, *new_pwd*, *new_pwd*	“Changing a group’s password” on page 124

Configuration quick-start

Note This section assumes that you are not enforcing security at TRS.

The following are brief, step-by-step instructions for configuring TRS. These steps help you run the sample programs described in Appendix B, “Testing a TRS Installation with Sample Programs” after you first install TRS.

❖ **To configure TRS**

1 Set the TRS Security property to no.

See “Security” on page 30.

2 Start TRS

DirectConnect for z/OS Option brings up TRS at start-up as long as the TRS exists in one of the following directories:

- For Windows:
  C:\<install_dir>\DC-15_0\servers\svclib\srvname
- For UNIX:
  /<install_dir>/DC-15_0/servers/srvname/svclib/

3 Start an isql session, connecting to TRS

4 Do one of the following:

- *LU 6.2 only:* Use sgw_addcon to define the connections your TRS uses.

  See “Adding a connection configuration” on page 48.

  ```
  exec sgw_addcon con_name, region, mode,
     "max_sessions"
  ```

- *TCP/IP only:* Use sgw_addregion to specify the regions your TRS uses.

  See “Defining regions to TRS” on page 50.

  ```
  exec sgw_addregion region, hostname,
    "port_number", regiontype
  ```

5 Use the sgw_addrpc procedure to add remote procedure calls (RPCs).

See “Defining RPCs to TRS” on page 52.

```
exec sgw_addrpc rpc_name, tran_id, region, security
```

The TRS client is now able to log in to TRS with a valid host user ID and password and execute the added RPCs.

Note Be sure to set up the default SQL language transaction as AMD2, SYRT, or SYIH. For more information, see "Defining a default SQL language handler" on page 55.

Configuring service communications

Use the instructions in this section if you are installing TRS, a new Mainframe Connect Server Option transaction, or if you need to configure TRS to use the Mainframe Connect DB2 UDB Option.

There are some differences in the steps required depending on whether you are using LU 6.2 connections or TCP/IP:

- For LU 6.2, configure TRS by defining mainframe *connections* and client RPCs to TRS.
- For TCP/IP, configure TRS by defining mainframe *regions* and client RPCs to TRS.

Configuring connections for LU 6.2

This section explains how to define new LU 6.2 connection configurations to TRS and how to remove existing connection configurations.

Every connection between TRS and a transaction processing region must be defined to TRS. When you execute an RPC, TRS chooses a connection with a *region* value that matches the definition in the RPC to execute the transaction.

Consider the following:

- If you are not using parallel sessions, do not use the same LU 6.2 pair or more than one session on it for more than one TRS, or for any other LU 6.2 application. Dedicate a separate set of connections for each TRS.

- If you use parallel sessions, you can use an LU 6.2 pair for TRS. However, be sure that you configure a sufficient number of sessions for the total number of Mainframe Connect Server Option users and Mainframe Connect Client Option users. Also, be sure that the workstation is configured as the "contention winner." (Check with your mainframe system programmer.)

Note When possible, Sybase recommends limiting use of an LU 6.2 pair to only one other TRS LU 6.2 application. This configuration simplifies the analysis if there are any LU 6.2 problems.

Adding a connection configuration

Add a connection configuration to TRS for each LU 6.2 pair defined to your SNA support. To define a new connection configuration to TRS:

```
exec sgw_addcon con_name, region, mode,
 "max_sessions"
```

where

- *con_name* is the name assigned to this local connection. It is also the name by which the *Local LU* is known to your local SNA support. Because there is a secondary name that qualifies this connection, this parameter corresponds to different values for different platforms. See the Mainframe Connect DirectConnect for z/OS Option *Installation Guide* for specific information about connection name parameter values.

  Length: maximum of eight characters.

- *region* specifies the *remote LU* name of the target mainframe region in this parameter. This is the Virtual Telecommunications Access Method (VTAM) APPLID name to which your *Local LU* is bound. An entry in this field is required.

  All RPCs that use this connection configuration to access the mainframe must have this same value specified as the *region* in their RPC definitions. (See also "Adding an RPC" on page 52.)

  Length: maximum of eight characters.

  For different platforms, this parameter corresponds to different values. See the Mainframe Connect DirectConnect for z/OS Option *Installation Guide* for specific information about the mode name parameter value.

- *mode* is a value that must match the name of the mode defined to the mainframe and to the local SNA support for this *LU* pair.

  Length: maximum of 8 characters.

- *"max_sessions"* is the maximum number of sessions that this TRS can have simultaneously allocated from the LU pair. Enter one of the following:

  - For parallel sessions, enter a value between 2 and 255.
  - For a single session, this value can only be 1.

  Be sure to enclose numeric parameter values in quotation marks.

  Check with your SNA System Administrator to make sure this number is not larger than the maximum number of sessions (for this mode) defined to the SNA subsystem.

Note If you do not provide a value for "max_sessions," TRS creates a default value = 1 for the connection, which will not support parallel sessions.

Example

This example adds an LU 6.2 connection configuration named "SYBLU01," bound to region (remote LU) "TESTREG," with mode name "M6S1024V," and not using parallel sessions:

```
exec sgw_addcon SYBLU01, TESTREG, M6S1024V, "1"
go
```

This example adds an LU 6.2 connection configuration named "SYBLU01," bound to region (remote LU) "PRODEMO," with mode name "M6P1024V" and eight parallel sessions:

```
exec sgw_addcon SYBLU01, PRODEMO, M6S1024V, "8"
go
```

Dropping a connection configuration

To delete all LU 6.2 connection configurations of a particular *con_name* from TRS:

```
exec sgw_dropcon con_name
```

where *con_name* with the name of the connection configuration you want to drop. The connection configuration name appears in the connections status display.

Example

To drop the connection configuration named “SYBLU01,” use the following procedure:

```
exec sgw_dropcon SYBLU01
go
```

Dropping individual regions from a connection configuration

To delete a connection configuration for a particular *LU* pair from a connection configuration:

```
exec sgw_dropcon con_name, region
```

Provide a value for the *con_name* parameter and the *region* (optional) parameter to drop a specific connection.

Warning! Providing only the *con_name* parameter value deletes all connection configurations for that *con_name*.

Example

This isql example deletes both sets of connection configurations added in the example:

```
exec sgw_dropcon SYBLU01
go
```

This example deletes only the eight connections defined to “PRODEMO”:

```
exec sgw_dropcon SYBLU01, PRODEMO
go
```

Configuring regions with TCP/IP

This section describes how to define and drop regions to TRS using TCP/IP.

Defining regions to TRS

For TRS to recognize the specified *region* parameter of the sgw_addrpc procedure, you must define the region using sgw_addregion:

```
exec sgw_addregion region, hostname, portnumber,
regiontype
```

where:

- *region* is a TRS administrator-defined alias for the *hostname* and *portnumber* pair, described next. For RPCs to use this region, this value must match the value in their region parameter of the sgw_addrpc procedure. (See "Adding an RPC" on page 52.)

  Length: maximum of eight characters.

- *hostname* is the value you specify for this parameter that identifies the TCP/IP network host name. This name corresponds to the mainframe in your */etc/hosts* file or in your NIS map.

  Length: maximum of 31 characters.

- *portnumber* is the number you specify for this parameter that must match the port number on which the transaction listens. (This is not the same as the port number used to configure the *interfaces* file.) TRS does not verify the validity of this number with the CICS TCP/IP Listener.

  This value can be any number between 1024 and 9996.

- *regiontype* (optional) is the type of the mainframe processing environment specified by the region parameter. Valid values are CICS, MVS, and IMS. If you do not specify a value, the region type defaults to CICS.

Dropping a region

To remove a *region* from those configured to TRS:

```
exec sgw_dropregion region
```

where *region* is the name of the region you intend to drop.

Configuring RPCs

A remote procedure call (RPC) is an Mainframe Connect Server Option mainframe application. TRS can be configured to invoke any Mainframe Connect Server Option mainframe application.

This section explains how to define new RPCs to TRS, and how to remove existing RPC definitions.

Defining RPCs to TRS

When TRS receives a request from a client, it needs the following information before it can forward the request to the mainframe:

- The name of the associated mainframe transaction
- The name of the *region* that identifies connectivity to the mainframe location where the transaction runs (defined in the sgw_addcon or sgw_addregion procedure)

You define this information to TRS when you add an RPC.

Adding an RPC

To define a new RPC to TRS for each new Open ServerConnect transaction and map it to a region:

```
exec sgw_addrpc rpc_name, tran_id, region,
 security
```

where:

- *rpc_name* is the TRS alias for the remote procedure. This is the name the client uses to call this RPC.

  Length: maximum of 30 characters.

- *tran_id* is the name by which the associated transaction is known on the mainframe. This is the mainframe transaction that TRS calls when a client requests the named procedure. The value of this field must be in uppercase. If the first character is numeric, the *tran_id* must be in quotes. Length:
  - For CICS: maximum of 4 characters
  - For IMS: maximum of 8 characters
  - For MVS: maximum of 8 characters
- *region*
  - For LU 6.2, specifies the *region* name used to identify the LU 6.2 connection configuration in the sgw_addcon procedure.

    At least one defined connection configuration must have this value specified as its region. See “Adding a connection configuration” on page 48. An entry in this field is required.

- For TCP/IP, specifies the *region* name used to identify this TCP/IP connection in the sgw_region procedure. See “Defining regions to TRS” on page 50. This field is required.
  Length: maximum of 8 characters.

- *region*
  (TCP/IP only) Specifies the *region* name used to identify this TCP/IP connection in the sgw_region procedure. See “Defining regions to TRS” on page 50. This field is required.
  Length: maximum of 8 characters.

- *security*
  Specifies the type of user login information that TRS passes to the mainframe. The *security* parameter is not case sensitive.

  - The *security* parameter can have any of these values to specify the information to send:

    none – do not send login information to the mainframe.

    userid – send only the user ID to the mainframe. To determine which user ID is used, see “Security level source” on page 54.

    Note This setting is not applicable when using the TRS TCP/IP Library.

    both – send both the user ID and the password to the mainframe. To determine which user ID is used, see “Security level source” on page 54.

  - If you are using LU 6.2, TRS passes the information in the conversation-level security fields of the SNA LU 6.2 Function Management Header 5 (FMH-5).

  - With TCP/IP, TRS passes these fields to the Listener Transaction when the called transaction starts.

  For example, if you use native CICS security, the none value corresponds to the CICS security option NONE, userid corresponds to IDENTIFY, and both corresponds to the security option VERIFY.

Note SNA network products vary in that some do not allow only the user ID to be forwarded; in other words, the *ALREADY VERIFIED* bit may not be set. Check your platform-specific DirectConnect and vendor SNA documentation for restrictions.

Example

To add an RPC named “SYD2”:

```
exec sgw_addrpc SYD2, SYD2, TESTREG, none
go
```

This maps SYD2 to the mainframe transaction named “SYD2,” which executes in the mainframe region named “TESTREG.” A user ID or password is not passed through to the mainframe when the RPC is invoked.

Security level source

When you invoke an RPC defined with a security parameter value of userid or both, the values passed to the mainframe for the user ID and password can come from one of three different pairs of values:

- If TRS security is off (see the security configuration parameter), TRS passes to the mainframe the user ID and password that is used to login to TRS.
- If TRS security is on, (see the security configuration parameter), and the *rpcpwd* level for the invoked RPC is defined as user (see sgw-addrpctogrp), TRS passes to the mainframe the user ID and password defined to TRS using the sgw_addlog procedure.
- If the security is on (see the security configuration parameter), and the *rpcpwd* level for the invoked RPC is defined as group (see sgw_addrpctogrp), TRS passes to the mainframe the user ID and password defined to TRS using sgw_addtrngrp for the *trn* group of that particular rpc.

For more information about these value sets, see sgw_addrpc.

Dropping an RPC

To drop an RPC, use this procedure (the RPC must be idle to be dropped):

```
exec sgw_droprpc rpc_name
```

where *rpc_name* is the name of the RPC you intend to drop.

Example

To drop the sample RPC named “SYD2”:

```
exec sgw_droprpc SYD2
go
```

Configuring a default SQL language handler for TRS

To pass client SQL language requests through TRS to the DB2 on the mainframe, the TRS System Administrator must configure a default SQL language handler. This language handler is a TRS RPC that is mapped to an Mainframe Connect Server Option program on the mainframe that handles the interaction with DB2 UDB.

Sybase provides these programs:

- For SQL language requests (including cursors, dynamic, and long-running transactions) on CICS, the Mainframe Connect DB2 UDB Option using the CICS transaction name AMD2
- For SQL language requests on IMS or MVS, OmniSQL Access Module for DB2 using the transaction name SYRT

Use the instructions in this section to define the default SQL language handler RPC to TRS by specifying the mainframe transaction ID that handles language requests.

Defining a default SQL language handler

The values you provide for defining the default SQL language handler to TRS depend on:

- The environment on the mainframe (CICS, IMS, or MVS)
- The status of TRS security (enforced or not enforced)

To define a default SQL language handler for your site configuration, use sgw_addrpc:

```
sgw_addrpc rpc_name, tran_id, region, security
```

Refer to Table 3-6 for the appropriate *rpc_name* and *tran_id* parameters, then set *region* and *security* to the appropriate values for your site:

Table 3-6: Default SQL language handler settings by host and security settings

TRS	CICS host (Mainframe Connect)	MVS host and IMS host (Mainframe Connect DB2 UDB Option)
TRS Security Enforced	• Set *rpc_name* = name you create • Set *tran_id* = AMD2	• Set *rpc_name* = name you create • Set *tran_id* = SYRT
TRS Security Not Enforced	• Set *rpc_name* = SYRT • Set *tran_id* = AMD2	• Set *rpc_name* = SYRT • Set *tran_id* = SYRT

Note If security is enforced, define the *rpc_name* that you create as the default SQL language handler when defining the *tran_group*. When security is *not* enforced, the default SQL language rpc_name must be SYRT.

Example

This is an isql example of the sgw_addrpc procedure, which defines a default SQL language handler:

```
exec sgw_addrpc SYRT, AMD2, TESTREG, both
go
```

where:

- Language requests are routed to the Mainframe Connect DB2 UDB Option, which has a predefined CICS *tran_id* of AMD2.
- TESTREG is the *region* parameter value that corresponds to the LU 6.2 or TCP/IP connection that provides access to the CICS region running the Mainframe Connect DB2 UDB Option.

If TRS security were enforced for this example, the TRS system administrator would need to define a transaction group (using sgw_addtrngrp) with SYRT as the default language RPC. SYRT then would become the default SQL language handler for all TRS users assigned to that transaction group.

Defining multiple SQL language handlers

To send SQL language requests to more than one language RPC (that is, if you have copies of the Mainframe Connect DB2 UDB Option for CICS and IMS in other regions), define multiple language handlers, each with a different RPC name and a different region name. Clients can explicitly specify the particular language RPC in the request.

Add the alternate language handlers using the sgw_addrpc procedure, using the following parameter values:

- An *rpc_name* of your choice
- The *tran_id* as defined in "Default SQL language handler settings by host and security settings" on page 56
- The *region* and *security* values appropriate for your site

This isql example shows three procedures that define different RPCs for the AMD2 transactions at the DALLAS, DETROIT, and MIAMI regions, respectively. This example allows a user to specify the region to which TRS sends the language request: "DALLASrpc" uses the DALLAS region, "DETROITrpc" uses the DETROIT region, and "MIAMIrpc" uses the MIAMI region.

```
exec sgw_addrpc DALLASrpc, AMD2, DALLAS, both
 go
exec sgw_addrpc DETROITrpc, AMD2, DETROIT, both
 go
exec sgw_addrpc MIAMIrpc, AMD2, MIAMI, both
 go
```

If security is enforced, add the preceding RPCs to the appropriate *tran_group* that the *tran_group* users can access:

```
exec sgw_addrpctogrp TRANGRP2, DALLASrpc, user
go
exec sgw_addrpctogrp TRANGRP2, DETROITrpc, user
go
exec sgw_addrpctogrp TRANGRP2, MIAMIrpc, user
go
```

To obtain results from "TESTREGG" with SYRT as the default (using the "Example" on page 56), a user invokes a SQL query program, such as isql, and enters a query similar to this:

```
select * from payroll
go
```

To obtain results from MIAMI, a user enters the execute command, specifying the RPC named "MIAMIrpc":

```
exec MIAMIrpc "select * from payroll"
go
```

Adaptive Server stored procedure example

This example shows how to create an ASE stored procedure that connects to TRS and uses a parameter to choose the DB2 UDB system to use:

```
create proc dbp1
              @salary int
            as
              if @salary < 60000
                    exec BLUETRS...MIAMIrpc "select *
from payroll"
            else
                    exec BLUETRS...DALLASrpc "select *
from payroll"
```

In this example, the Adaptive Server is configured to connect to the TRS named “BLUETRS” and to execute an RPC. The value of the @salary parameter determines the language RPC that “BLUETRS” uses to route the select statement. Based on the value of @salary, one of the following occurs:

- If the value of @salary is less than 60,000, the procedure sends the select * from payroll statement to the TRS named “BLUETRS.”
  In “BLUETRS,” the AMD2 transaction in MIAMI executes it.
  The AMD2 transaction in MIAMI is mapped to the RPC named “MIAMIrpc” on “BLUETRS,” or
- If the value of @salary is greater than or equal to 60,000, the procedure executes against the AMD2 transaction in DALLAS, which is mapped to the RPC named “DALLASrpc” on “BLUETRS.”

Using Catalog Stored Procedures (CSPs)

CSPs serve as a uniform catalog interface for accessing the system tables of different database management systems, including Adaptive Server.

Sybase provides CSPs that can be defined to TRS with the Mainframe Connect DB2 UDB Option. These CSPs correspond to transactions on the mainframe that access the DB2 UDB catalog and return information to a client application in a standard format.

CSPs are implemented as CICS transactions and must be configured as RPCs to TRS. See Chapter 4, “Accessing Catalog Information with CSPs,” for configuration instructions.

Table 3-7 outlines the functions provided by CSPs and the 4-character CICS transaction name that each procedure name maps to on the host:

Table 3-7: CSP functions

Procedure name	Related CICS transaction name	Function
sp_capabilities	SYSP	Returns information about the capabilities of the TRS
sp_columns	SYSP	Describes the columns of a table
sp_column_privileges	SYSP	Describes the column permissions of an object
sp_databases	SYSP	Lists the databases available
sp_datatype_info	SYCR	Describes the datatypes available
sp_fkeys	SYSP	Describes the primary key/foreign key relationships
sp_pkeys	SYSP	Describes the primary key for a table
sp_server_info	SYSP	Lists the configuration and capabilities
sp_special_columns	SYSP	Lists the optimal columns to uniquely identify rows and list columns that are automatically updated
sp_sproc_columns	SYSP	Describes the input/output of executable objects
sp_statistics	SYSP	Lists the index and statistics information about a table
sp_stored_procedures	SYSP	Lists the executable procedures
sp_table_privileges	SYSP	Describes the *table permissions*
sp_tables	SYSP	Lists the tables
sp_thread_props	SYCR	Returns minimal information at this time

For information about the mainframe installation, see the Mainframe Connect *Installation and Administration Guide* for DB2 UDB. For information about the syntax and operation of CSPs, see Chapter 4, “Accessing Catalog Information with CSPs,” in this guide.

CSP scripts

Sybase provides three scripts for you to use with CSPs:

- addcat - adds the CSPs to TRS.

- dropcat - drops the CSPs from TRS.
- testcat - tests the CSPs (requires that the AMD2 transaction be installed at the mainframe).

Installing CSPs

The addcat script executes the sgw_addrpc procedure automatically for each CSP (see “Adding an RPC” on page 52). Before you run addcat, modify the script to suit your installation.

Use your text editor to specify the value of these parameters:

- *region* parameter – name of the region you want the CSPs to execute against.
- *security* parameter – value you can change to meet the security requirements at your installation. If you do not change it, the value is none.
- *rpc_name* parameter – name or value must be coordinated with any change to the RPC names with the mainframe system programmer. If you are using ODBC applications, do not change the RPC names.
- *tran_id* parameter – value or name of this parameter must be coordinated with any change to the transaction ID with the mainframe system programmer.

After you edit the script to suit your installation, run the addcat script as input to your TRS. The following isql example shows how to run the addcat script with a TRS named “new_TRS”:

```
isql -Snew_TRS -Usa -P < addcat
go
```

This script automatically executes the sgw_addrpc procedure for each CSP.

Testing CSPs

The testcat script uses the AMD2 transaction to create temporary tables and execute each CSP. At least one row is returned for each CSP, and the testcat script then drops the temporary tables.

Example

Run the testcat script as input to your TRS. This isql example shows how to run the testcat script with a TRS named “new_TRS”:

```
isql -Snew_TRS -Usa -P < testcat
```

```
go
```

This script automatically tests each of the CSPs.

Dropping CSPs

The dropcat script drops the CSPs from TRS. Run the dropcat script as input to your TRS.

Example

This isql example shows how to run the dropcat script with a TRS named "new_TRS":

```
isql -Snew_TRS -Usa -P < dropcat
go
```

This script automatically drops the CSPs.

CHAPTER 4

Accessing Catalog Information with CSPs

Using CSPs

To obtain information about database objects, you need to access the database catalog. Catalog Stored Procedures (CSPs) provide this catalog access.This section describes the use of CSPs, coding instructions that apply to CSPs, and the use of wildcard-character search patterns.

Why use CSPs?

The catalog structures for DB2 UDB and ASE are different. If you have client applications written to access the SQL Server catalog, you may need to re-code the client application queries to send those queries directly to the DB2 UDB 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).

You need to install and configure CSPs in TRS for the proper functioning of the following clients:

- ODBC
- ASE/CIS

Coding instructions

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

Parameters

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

These rules apply to CSP 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. However, when using CSPs, these object names must be referred to in uppercase.

Syntax

A client application can initiate a CSP by issuing one of these statements:

```
exec rpc_name parm1, parm2, . . .
```

```
execute rpc_name parm1, parm2, . . .
```

where:

- *rpc_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.

To specify the parameters for a CSP, use 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:

  ```
  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 can use the named parameter form if you process a CSP as an RPC event.

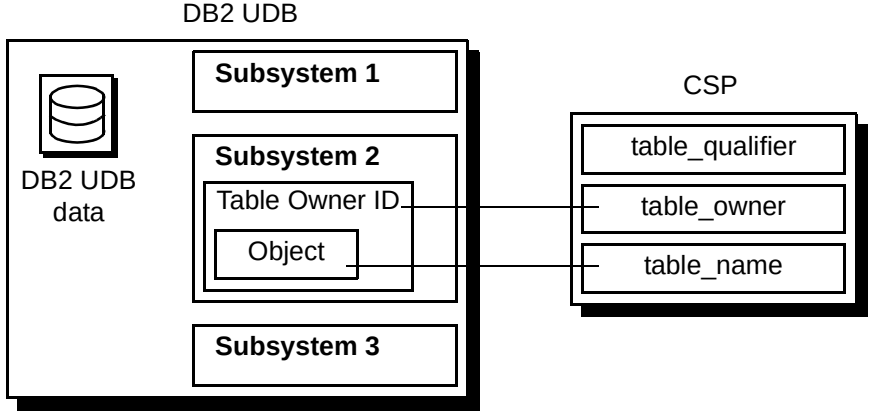
Table name, owner, and qualifier parameters

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.

Figure 4-1 shows how CSP parameters relate to the DB2 UDB subsystem:

Figure 4-1: Relationship of CSP parameters and DB2 UDB



Wildcard-character search patterns

The percent (%) wildcard character can be used in parameters that allow wildcard-character search patterns. This wildcard represents any string of zero or more characters. When using CSPs, the wildcard expression must be enclosed in quotation marks.

If the percent (%) character is used in parameters that do not allow wildcard-character search patterns, you receive a syntax error.

Table 4-1 shows some examples of the percent (%) wildcard character and its use:

Table 4-1: Wild card character 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 (@ and %) 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.

Supported CSPs

Table 4-2 shows the supported CSPs and the information that each CSP retrieves:

Table 4-2: 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

The following sections provide descriptions, syntax, parameters, and usage for the supported CSPs.

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 an DB2 access service during normal query processing.

Results

Table 4-3 shows the result set:

Table 4-3: 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

Table 4-4 shows the ID and values for several DB2 access service functional capabilities:

Table 4-4: 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

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 and 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.

Table 4-5 shows the result set for sp_column_privileges:

Table 4-5: 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.

Table 4-6 shows the result set for sp_columns:

Table 4-6: 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
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 SQL 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

Table 4-7 describes the DB2 UDB 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 4-7: ODBC datatypes

DB2 UDB datatype	Target datatype maximum physical length	ODBC type	ODBC integer ID	DB2 UDB 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
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

DB2 UDB datatype	Target datatype maximum physical length	ODBC type	ODBC integer ID	DB2 UDB datatype description
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. Table 4-8 describes the datatype value.

Table 4-8: 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 Bits 24–31	 Precision 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

None.

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.

Table 4-9 shows the result set for sp_databases.

Table 4-9: 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, otherwise 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 4-7 on page 76 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 lengths for varchar columns shown in the result set tables are maximums; the actual lengths depend on the target database. Table 4-10 shows the result set for sp_datatype_info.

Table 4-10: 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 used to prefix a literal; NULL is returned for datatypes where a literal prefix is not applicable.
LITERAL_SUFFIX	varchar(128)	Character 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 – the column does not accept NULL values. • 1 – the column accepts NULL values.
CASE_SENSITIVE	smallint NOT NULL	Indicates whether the datatype distinguishes between uppercase and lowercase characters: • 0 – the datatype *is not* a character type or is not case sensitive. • 1 – the datatype *is* a character type and is case sensitive.
SEARCHABLE	smallint NOT NULL	Indicates how this datatype is used in where clauses: • 0 – the datatype cannot be used in a where clause. • 1 – the datatype can be used in a where clause.

Column	Datatype	Description
UNSIGNED_ATTRIBUTE	smallint	Indicates whether this attribute is unsigned: • 0 – the datatype is signed. • 1 – the datatype is unsigned. • NULL – the datatype is not numeric.
MONEY	smallint NOT NULL	Indicates whether this is a money datatype: • 0 – it is not a money datatype. • 1 – it is a money datatype.
AUTO_INCREMENT	smallint	Indicates whether this datatype automatically increments: • 0 – columns of this datatype do not automatically increment. • 1 – columns of this datatype automatically increment. • NULL – 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.

pktable_owner
is the owner of the table containing the primary 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.

pktable_qualifier
is ignored. Leave blank or set to NULL.

fktable_name
is the name of the table containing the foreign key. Views, aliases, and wildcard-character search patterns are not supported.

fktable_owner
is the owner of the table containing the foreign 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.

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.

Table 4-11 shows the result set for sp_fkeys:

Table 4-11: 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.

Table 4-12 shows the result set for sp_pkey:

Table 4-12: 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.

Table 4-13 shows the result set for sp_server_info:

Table 4-13: 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.

table_owner
is the owner of the table. 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.

Table 4-14 shows the result set for sp_special_columns.

Table 4-14: 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 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.

Table 4-15 shows the result set for sp_sproc_columns.

Table 4-15: 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_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.

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.

Table 4-16 shows the result set for sp_statistics.

Table 4-16: 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
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 – statistics for a table • 1 SQL_INDEX_CLUSTERED – a clustered index • 2 SQL_INDEX_HASHED – a hashed index • 3 SQL_INDEX_OTHER – 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 – ascending • D – descending • NULL – 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

Column	Datatype	Description
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

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.

Table 4-17 describes the result set for sp_stored_procedures:

Table 4-17: Result set for sp_stored_procedures

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

Table 4-18 shows the result set for sp_table_privileges:

Table 4-18: 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

Column name	Datatype	Notes
PRIVILEGE	varchar (128) NOT NULL	Identifies the privilege granted to the grantee on this object as one of the following values: • SELECT – the grantee is authorized to select rows in the associated object. • INSERT – the grantee is authorized to insert rows into the associated object. • UPDATE – the grantee is authorized to update rows in the associated object. • REFERENCES – 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 – the grantee can grant this privilege to others. • NO – the grantee cannot grant this privilege to others. • NULL – 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

- 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.

Table 4-19 shows the result set for sp_tables:

Table 4-19: 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

sp_thread_props

Description — Allows the client to retrieve and set various thread properties.

Syntax — sp_thread_props [*property_name* [, *property_value*]]

Parameters

property_name
is the name of the property to be set or shown.

property_value
is the value to which the property is to be set.

Usage — If you do not provide any parameters, or if you provide only *property_name*, the access service returns a single result set consisting of every instance of *property_name* and the value for each.

CHAPTER 5

Configuring a TRS Library for Security

Note If you do not enforce security at TRS (that is, if you set the TRS Security configuration property to no), there are still topics in this chapter that may be helpful. For example, if you defined RPCs to send a user ID and password to your mainframe security, the mainframe must recognize the user ID and password, even if you set the Security property to no.

Security overview

This chapter explains how to configure TRS to control client access to the following:

- TRS
- Specific host connections
- Mainframe transactions

TRS provides client access security, identifies security considerations and responsibilities, and uses existing security procedures to enforce security.

Security features

You can restrict client access to a mainframe processing environment in the following ways:

- Require client identification for access to TRS. Only clients specifically defined to a TRS are allowed to send requests through TRS.
- (*For LU 6.2 only*) Restrict access to mainframe connections. Each client login is assigned a group of connections it can use.
- Restrict access to mainframe transactions. Each client login is assigned a group of permitted mainframe transactions.

By default, TRS security is automatically enabled when you start TRS. You can specifically override it by setting the TRS Security configuration property to no.

Security considerations

When you plan security, you must consider security requirements at each of the following network nodes: client, TRS, and mainframe. Your security plan for TRS must address the following issues:

- *Client permissions.* Does the client have permission to log in to the network? Can the client's login information be stored and passed along to TRS for permission checking at that level? Can it be passed to the mainframe to support security systems in use there?
- *Adaptive Server permissions*. If client requests are routed to TRS through Adaptive Server, which commands, data objects, stored procedures, and views does the client have permission to use? Will the client use long-running transactions? (Long-running transactions can be sent through ASE/CIS.)
- *(LU 6.2 only) Mainframe connection permission.* Does the client have permission to use a given LU 6.2 connection to the mainframe?
- *Mainframe transaction permission.* Does the client have permission to execute a given mainframe transaction?
- *Mainframe data resource permission.* Does the client have permission to access or modify the data in a particular file or database?

For information on client login and ASE security, see the ASE and Open Client and Open Server documentation. Sybase security at the mainframe is described in the Open ServerConnect *Installation and Administration Guide for IBM CICS/MVS* and Mainframe Connect DB2 UDB Option *Installation and Administration Guide*. You can find additional information about mainframe security in your vendor documentation.

Security responsibilities

Each instance of a TRS LU62 or TRS TCP/IP Library has its own responsibilities for security. The following section discusses security responsibilities peripheral to TRS.

Client workstation

Most sites on the network have a secure login procedure that verifies the user's identity and authorization by requiring a unique user ID and password. The user ID, password, and profile information can be passed to ASE and to TRS.

Adaptive Server

Adaptive Server can grant or deny a user permission to call a particular remote procedure. Requests routed to TRS through ASE undergo security checks. The TRS administrator can apply this security mechanism to all TRS requests by setting the TRS DirectPrevent configuration property to yes, which requires all client requests to pass through ASE before they are routed to TRS.

There are two ways to get to TRS from ASE:

- ASE/CIS
- ASE site handler

Network level

The vendor's SNA support software allows login information to be sent to the mainframe in FMH-5 fields along with client requests. This facility allows you to use external security products that require client login information.

TCP/IP sends login information to the CICS Listener Transaction when the CICS transaction starts.

Security quick-start

Here are brief, step-by-step instructions for setting up security for TRS. This section assumes that mainframe security is already configured to match the values you will specify as you go through these steps. For details, see the complete description of each procedure that follows in this chapter.

❖ **To set up security for TRS**

1 Set the TRS Security configuration property to yes

2 Start TRS.

3 Assign a password to the "sa" account.
(See "Changing user passwords and logins" on page 106.)

```
exec sgw_chpwd sa, password
```

Note Remember this password. If you forget passwords for all TRS logins with administration privileges, you will have to reconfigure all of TRS security.

4 *(LU 6.2 only)* Use sgw_addcon to define the connections your TRS uses. Specify LUs that use a mode entry that supports conversation level security. Talk to your VTAM system programmer and verify that the PSERVIC property has a value of "x'12'" or "x'10'" in the tenth byte.

```
exec sgw_addcon con_name, region, mode,
 "max_sessions"
```

See "Adding a connection configuration" on page 48.

5 (*LU 6.2 only*) Use sgw_addcongrp to add a connection group:

```
exec sgw_addcongrp group_name
```

See "Adding a connection group" on page 110.

6 For LU 6.2 or TCP/IP:

- (*LU 6.2 only*) Use sgw_addcontogrp to add connections to the connection group:

  ```
  exec sgw_addcontogrp group_name, con_name
  ```

  See "Adding connections to a connection group" on page 110.

- *(TCP/IP only)* Use sgw_addregion to specify the regions:

  ```
  exec sgw_addregion region, hostname,
  "port_number"
  ```

See “Defining regions to TRS” on page 50.

7 Use sgw_addrpc to add RPCs:

```
exec sgw_addrpc rpc_name, tran_id, region, security
```

In the sgw_addrpc procedure, use one of these *security* parameters to specify the login information to send to the mainframe for each RPC:

- none – do not send login information to the mainframe.
- userid – send only the user ID to the mainframe.
- both – send both the user ID and the password to the mainframe. (Use values that your mainframe security recognizes.)

See “Adding an RPC” on page 52.

8 Use the sgw_addtrngrp procedure to add a transaction group:

```
exec sgw_addtrngrp tran_group, GROUP_LOGIN,
 GROUP_PWD, langrpc, langpwdlevel
```

See “Adding a transaction group” on page 114.

Note The values of GROUP_LOGIN and GROUP_PWD must be uppercase.

9 Use sgw_addrpctogrp to add RPCs to the transaction group:

```
exec sgw_addrpctogrp tran_group, rpc_name,
 rpcpwdlevel
```

For each RPC you add to the group, specify the source of the mainframe login using one of these rpcpwdlevel parameters:

- none – do not send login information to the mainframe.
- user – send the host login and password specified in the sgw_addlog procedure (see the next step) to the mainframe.
- group – send the login and password specified in the sgw_addtrngrp procedure (see “Adding a transaction group” on page 114) to the mainframe.

See “Adding RPCs to a transaction group” on page 116.

10 Use sgw_addlog to add a login. Specifying the transaction group and connection group that you added in the previous steps:

```
exec sgw_addlog login, pwd, HOST_LOGIN, HOST_PWD,
tran_group, con_group, gwctrl
```

See “Adding a login” on page 105.

Note Be sure the values of HOST_LOGIN and HOST_PWD are in uppercase. For LU 6.2, use the con_group parameter. For TCP/IP, include a comma as a placeholder.

TRS Administrator security tasks

Under TRS security, every client login must be defined to TRS. This login definition specifies the client login ID and password, as well as an optional mainframe login ID and password for each. A login definition also includes an assignment to a connection group (LU 6.2 only) and a mainframe transaction group. Clients using that login can only access connections and transactions in their assigned groups.

A transaction group lists RPCs that are defined to TRS. Each RPC in the group corresponds to a specific mainframe transaction. When a client calls a remote procedure, the corresponding mainframe transaction executes.

Basic responsibilities of the TRS Administrator are outlined in “Security quick-start” on page 100, which is an overview of steps to set up TRS security.

Overriding security

If you do not want to enforce security at TRS, you can disable TRS security by setting the TRS Security=no. This option tells TRS not to verify logins (except for “sa”) or to allow access to verify transactions and connections.

When you set Security=no, user IDs and passwords used to log in to TRS are forwarded transparently to the mainframe on each RPC. This method uses mainframe security only. See “Adding an RPC” on page 52 for information about RPC security definitions.

User IDs

When you enforce security at TRS, you can choose to assign a single mainframe ID to all clients that use a certain transaction or group of transactions rather than have all individual user IDs and passwords defined. This group ID is specified as part of the transaction group definition with the sgw_addtrngrp procedure. See “Adding a transaction group” on page 114 for more information.

System Administrator’s account

When first installed, TRS has a single client login defined as “sa” (system administrator). This login has permission to use all control and security features of TRS. Initially, a password is not required to log in as “sa.” You should define your own password for the “sa” login as soon as you begin setting up TRS.

To change the password:

```
exec sgw_chpwd login, gateway_pwd, HOST_PWD
```

- Replace login with “sa,” and gateway_pwd with the password for TRS.
- You can omit the HOST_PWD parameter unless you defined the “sa” account at the mainframe as well.
- You do not need to include the comma as a placeholder because it is the last parameter in the procedure.
- If you include a password for the transaction processing region at the mainframe (host), enter it in uppercase.

For more information, see “Changing user passwords and logins” on page 106.

Note Remember the password of the TRS “sa.” If you forget the passwords for all TRS logins with administrator privileges, you will have to reconfigure security.

Defining logins to TRS

When TRS security is enabled, a login definition must be defined for every client that wants to access TRS. This definition includes the login ID and password and groups of transactions and connections (LU 6.2 only) that are available to clients using this login.

When you define a login to TRS, you can specify a mainframe ID and password for that login. This feature enables a TRS client attempting to access mainframe resources to use IDs and passwords that the mainframe recognizes.

If security is enforced at TRS, when TRS receives a client request, it checks the client's login ID and password against its list of login definitions. If the client's login information matches a login definition entry, TRS accepts the login request. If it does not recognize the login information, it rejects the request. Only clients with IDs defined to TRS are allowed to log in to TRS.

See "Adding a login" on page 105 for more information about defining a login.

User-level security

When security is enforced at TRS (Security=yes), every user who sends requests to a transaction processing region through TRS must be defined to that TRS. A user definition includes this information:

- The user's login ID and password
- The transaction processing region (host) login ID and password in uppercase
- (*LU 6.2 only*) The assigned connection group that the user is permitted to use to access a mainframe
- The assigned transaction group defining the collection of RPCs the user is permitted to use
- The permission to perform TRS control operations

Displaying current logins

To display a summary of all existing logins:

```
exec sgw_dsplog
```

The sgw_dsplog procedure displays the login and host login name, the transaction group name, the connection group name (LU 6.2 only), and indicates whether the login can access the control procedures. All users can execute the status procedures.

Adding a login

To add a login definition to TRS:

```
exec sgw_addlog login, pwd, HOST_LOGIN,
  HOST_PWD, tran_group, con_group, gwctrl
```

where:

- *login* is the login ID of the user, sent from the client application.
  For example, this would be the value provided in the -U flag specified in isql.
  Length: maximum of 30 characters.
- *pwd* is the login password.
- *HOST_LOGIN* is the login ID by which this user is known to the mainframe. Leave this field blank only if you are also not specifying a HOST_PWD. The value for this field must be in uppercase.
  Length: maximum of 8 characters.
- *HOST_PWD* is the password for the HOST_LOGIN. The value for this field must be in uppercase. Leave this field blank only if you are also not specifying a HOST_LOGIN.
  Length: maximum of 8 characters.
- *tran_group* is the name of the collection of RPCs this user can access. This collection must be defined to TRS, and a user can be assigned to only one transaction group (see “Adding a connection group” on page 110).
  Length: maximum of 8 characters.
- *con_ group* (LU 6.2 only) is the name of the collection of connections this user can access. This connection group must be defined to TRS, and a user can be assigned only one connection group. (See “Adding a connection group” on page 110.)

  For TCP/IP only, include a comma or null as a placeholder, but do not provide a value for the con_group parameter.
  Length: maximum of 8 characters.
- *gwctrl* is the TRS administration procedures permission indicator. Choose one of these values:

- yes grants the user permission to access and make changes using control, configuration, and security procedures.
- no means the user has status-querying permission only.

Note If you type something other than yes or no, the gwctl parameter defaults to no.

Example

To add the user named “BERTHA” to an LU 6.2 TRS:

```
exec sgw_addlog bertha, BIGBLUE, BIG, BLEUBRT, TGROUP1,
FINANCE, yes

go
```

This isql example adds a TRS user named “BERTHA” with a password of “BIGBLUE,” and a host login and password of “BIG” and “BLEUBRT,” respectively. “**B**ERTHA” can use RPCs defined to the transaction group named “TGROUP1” and connections included in the connection group named “FINANCE.” “BERTHA” has permission to administer TRS.

Changing user passwords and logins

Users can change their own passwords. Users with control authority can change other users’ passwords. (Control authority is defined by a yes value for the gwctrl parameter of the sgw_addlog procedure.)

Changing passwords

To change the TRS password or the TRS record of this user’s password for a login:

```
exec sgw_chpwd login, pwd, HOST_PWD
```

where:

- *login* is the name of the TRS login for which you intend to change the password.
- *pwd* is the password for TRS.

- *HOST_PWD* is the password for the mainframe. The value for this parameter must be in uppercase.

Note If you do not have a value for a parameter, (that is, if you only want to change one password) include the comma or null as a placeholder.

Example

To change the mainframe password for a user named "BERTHA" and keep the same TRS password:

```
exec sgw_chpwd BERTHA,null,BLUEBRT
go
```

The TRS password "BERTHA" is unchanged, and her new mainframe password is "BLUEBRT."

Changing logins

To change a user's login ID for TRS or for the mainframe (the HOST_LOGIN parameter of the sgw_addlog procedure), drop the login and add it again with the new ID.

For information about dropping a login, see "Deleting a user definition" on page 107.

Deleting a user definition

You can remove user definitions from the TRS list of logins. To delete a user from the list:

```
exec sgw_droplog login
```

where *login* is the TRS login name of the user you intend to drop.

Example

To remove the user named "BERTHA" from the TRS list of logins:

```
exec sgw_droplog BERTHA
go
```

Conversation-level security

For LU 6.2, conversation-level security occurs when TRS passes client login information to the mainframe in the conversation-level security fields of the Function Management Header (FMH)-5 along with the client's request. The mainframe uses this login information to determine whether the client has permission to use the requested resources. For TCP/IP, TRS sends the user ID and password to the Sybase Listener Transaction when the transaction starts.

When configuring RPCs and TRS security, you need to make decisions about:

- When to pass login information to the mainframe
- What login information to pass to the mainframe

When to forward login information

The mainframe may or may not require a full user ID and password complement for every requested transaction. When defining an RPC to TRS even when security is not enabled, you can specify the level of security information that best matches its mainframe component. Your choices are:

- none – TRS passes the request to the mainframe without any user ID or password.
- userid – TRS passes the user ID to the mainframe along with the request.
- both (user ID and password) – TRS passes the user ID and the password to the mainframe along with the request.

What login information to forward

Because user ID and password requirements at the mainframe can be different from those at the client workstation, you can specify a separate mainframe ID and password in the login definition (these values must be in uppercase). When mainframe values are specified, TRS forwards these mainframe values with the client request. If mainframe values are not specified, TRS does not forward the login information.

You can specify an alternate mainframe ID and password for a transaction group. When you add a transaction to the group, specify whether the login definition ID and password or the transaction group ID and password are passed to the mainframe with requests for that transaction.

Connection-level security (LU 6.2 only)

When connection-level security is enforced at TRS (Security=yes), a user must have explicit permission to use a particular host connection. You assign a connection group to each user defined to TRS. A connection group is a list of connections that are defined to your SNA support and TRS.

Connection groups

Assigning a connection group to a user gives that user permission to use any connection belonging to that group. A user can belong to only one connection group and can use only the connections in that group. If a user login definition does not have a connection group assigned to it, and that user sends a request when security is enforced at TRS, then TRS rejects that request.

Use the connection group procedures to:

- Define the connections that make up a connection group
- Modify that list by adding or deleting connections
- Query connection groups to determine the connections that belong to them
- Add or delete entire connection groups

All connections listed in a connection group must be defined to TRS and to your SNA support. When a connection is defined, you can assign it to any number of connection groups. Likewise, you can assign a connection group to any number of users.

Connection-level security allows you to:

- Dedicate a single specific connection to a particular user. To do this, define a connection group to include a single connection, then assign that connection group to a single user.
- Dedicate a group of connections to a particular user. To do this, define a connection group to include the desired connections, then assign that connection group to a single user.
- Dedicate a group of connections to a specific group of users. To do this, define a connection group to include the desired connections, then assign that connection group to all users in the group.

To add new connection groups to TRS and to modify and delete existing connection groups, use the procedures described in the following sections.

Displaying current connection groups

To display all connection groups currently defined to TRS:

```
exec sgw_dspcongrp
```

Displaying one connection group

To display detail about a particular connection group:

```
exec sgw_dspcongrp group_name
```

where *group_name* is the name of a connection group you want to display.

Example

```
exec sgw_dspcongrp FINANCE
go
```

This procedure returns a list of the connections in the connection group named "FINANCE."

Adding a connection group

To define a new connection group:

```
exec sgw_addcongrp group_name
```

where *group_name* is the name of the connection group you intend to add. The connection group name can be a maximum of 8 characters.

Example

To add the "FINANCE" connection group:

```
exec sgw_addcongrp FINANCE
go
```

Add connections to the new group as shown in the next section.

Adding connections to a connection group

After you add the new connection group, specify the connections that belong to it. For each connection you add:

```
exec sgw_addcontogrp group_name, con_name
```

where:

- *group_name* is the name of the connection group to which you intend to add a connection.
- *con_name* is the name of the connection you intend to add.

Re-execute sgw_addcontogrp for each connection you want to add to the group.

Example

To add the connection named “SYBLU01” to the “FINANCE” connection group:

```
exec sgw_addcontogrp FINANCE, SYBLU01
go
```

Dropping connections from a connection group

To remove connections from a connection group:

```
exec sgw_dropconfromgrp group_name, con_name
```

where:

- *group_name* is the name of the connection group from which you intend to drop a connection.
- *con_name* is the name of the connection you intend to drop.

Example

To delete the connection named “SYBLU01” from the connection group named “FINANCE”:

```
exec sgw_dropconfromgrp FINANCE, SYBLU01
go
```

Dropping a connection group

To delete an existing connection group:

```
exec sgw_dropcongrp group_name
```

where *group_name* is the name of the connection group you intend to drop.

Transaction-level security

When security is enforced at TRS (the Security=yes), a user must have explicit permission to use a particular RPC. To grant a user access to an RPC, assign a transaction group to the user’s login in the sgw_addlog procedure.

Assigning transaction groups

A transaction group is a collection of RPCs defined to TRS. Assigning a transaction group to a user gives that user permission to invoke a remote procedure, causing the corresponding mainframe transaction to execute.

Some considerations:

- A user can belong to only one transaction group and can execute only the transactions in that group.
- If a user request specifies an RPC that is not included in the user's transaction group, TRS rejects the request and returns an error message to the user.
- A transaction group can include any number of RPC names. It can also include one RPC name for which the associated mainframe transaction processes SQL language requests dynamically, called the language RPC.
- An RPC can exist in many transaction groups.

Defining a default SQL language handler

If you do not enforce security at TRS, the default RPC name for a SQL language handler is SYRT. To define the SYRT RPC to TRS, use the sgw_addrpc procedure. If security is enforced at TRS, a default language RPC name does not exist.

See "Adding an RPC" on page 52 and "Configuring a default SQL language handler for TRS" on page 55 for more information.

Defining group logins

Each user login has an associated mainframe login user ID and password, which are passed to the transaction processing region along with the client request. You can override this login for certain client requests with a group login that applies to all users who are assigned to the same transaction group. A group login and its password is defined when the transaction group is defined. (See "Adding a transaction group" on page 114.)

Specifying login ID levels

When you add a transaction to a transaction group, you must specify the login ID level passed to the transaction processing region whenever that transaction is requested:

- user – the user's transaction processing region login information.

- group – the transaction group login information.
- none – no login information.

The transaction group login allows you to use a single transaction processing region login for multiple users (for example, everyone in the Accounts Receivable Department).

Transaction group procedures

The transaction group administration procedures allow you to:

- Define the list of RPCs that belong to a group
- Modify that list by adding and deleting RPCs
- Add or delete entire transaction groups
- Specify a group login for the transaction group
- Specify the login, if any, to pass to the transaction processing region with a request
- List the following information about a transaction group:
  - The RPCs that belong to the group
  - The language transaction used by its users
  - The transaction processing region login information this group uses

All RPC names listed in a transaction group must be defined to TRS. They must map to transactions the names of which RPCs are defined to the mainframe transaction processing region.

After you define an RPC, you can assign it to any number of transaction groups. Also, you can define a transaction group to any number of users; however, each user can be associated with only one transaction group.

Displaying all transaction groups

You can add new transaction groups to TRS and modify and delete them. To display information about existing groups:

```
exec sgw_dsptrngrp
```

The sgw_dsptrngrp procedure, when entered without parameters, displays *all* transaction groups.

Displaying one transaction group

To display details about a particular transaction group:

```
exec sgw_dsptrngrp tran_group, rpc
```

where:

- *tran_group* is the name of the transaction group you want to display (see the following example).
- rpc is a keyword that you enter as a fixed-string, optional parameter to only display the RPCs that are members of that transaction group and the RPC password levels. If you omit rpc, the member RPCs are not included in the results.

Example

```
exec sgw_dsptrngrp TGROUP1
go
```

The results of this procedure list the following information:

- Group name
- Group login
- Group password
- Language handler
- Language password source

To list only the RPC name and the RPC password source, you can include the optional rpc fixed-string parameter:

```
exec sgw_dsptrngrp TGROUP1, rpc
go
```

Adding a transaction group

To define a new transaction group (replace the italicized parameters as shown):

```
exec sgw_addtrngrp tran_group, GROUP_LOGIN
 GROUP_PWD, langrpc, langpwdlevel
```

where:

- *tran_group* is the name of the transaction group.
  Length: maximum of 8 characters.

- *GROUP_LOGIN* is the alternate transaction processing region login that member transactions can use. When langpwdlevel is set to group, the GROUP_LOGIN overrides the HOST_LOGIN of the user calling this procedure. This value must be in uppercase. Null is valid.
  Length: maximum of 8 characters.
- *GROUP_PWD* is the alternate transaction processing region password that member transactions can use. When langpwdlevel is set to group, this password overrides the HOST_LOGIN of the user calling this procedure. This value must be in uppercase. Null is valid for TRS LU62 only.
  Length: maximum of 8 characters.
- *langrpc* is the RPC name used to process SQL language requests. This is the name assigned to all language requests by users of this transaction group. Null is valid.
  Length: maximum of 30 characters.
- *langpwdlevel* is the source of the transaction processing region login information for language RPCs. It indicates whether transaction processing region login ID and password should be passed to the transaction processing region with this transaction request, and if so, whether the user's HOST_LOGIN or the transaction group's GROUP_LOGIN information should be used. This parameter can have one of the following values:
  - none – do not send login information to the transaction processing region.
  - user – send the user's HOST_LOGIN and HOST_PWD.
  - group – send the GROUP_LOGIN and GROUP_PWD defined here.

Example

This example creates the transaction group named "TGROUP1":

```
exec sgw_addtrngrp TGROUP1,,,AMD2,user
go
```

This example gives the "TGROUP1" transaction group these characteristics:

- It does not use group logins or passwords.
- It uses the AMD2 language RPC.
- It forwards the HOST_LOGIN and HOST_PWD information of the users assigned to this group (in the sgw_addlog procedure) to the transaction processing region.

Adding RPCs to a transaction group

After you define a transaction group, you must specify the transactions that belong to it. A transaction group contains one language RPC and any number of standard RPCs.

To add an RPC to the transaction group:

```
exec sgw_addrpctogrp tran_group, rpc_name,
  rpcpwdlevel
```

where:

- *tran_group* is the name of the transaction group to which you want to add an RPC.
  Length: maximum of 8 characters.
- *rpc_name* is the name of the RPC you want to add. This is the remote procedure called by the client.
  Length: maximum of 30 characters.
- *rpcpwdlevel* indicates whether user identification is passed to the transaction processing region with this transaction request and, if user identification is to be passed, indicates the origin of the identification. This parameter can have one of the following values:
  - none – do not send login information to the transaction processing region.
  - user – use the user ID and password from the HOST_LOGIN and HOST_PWD values of the user login definition.
  - group – use the user ID and password from the GROUP_LOGIN and GROUP_PWD values of the transaction group definition.

Specifying an RPC password level

Specify one of the following IDs to send to the mainframe with the request:

- The group ID for the transaction group, defined using the sgw_addtrngrp procedure.
- The client's mainframe login and password from the client's login definition (userid), defined using the sgw_addlog procedure.
- none, which indicates that login information should not be sent to the mainframe with that transaction. In combination with setting the TRS Security configuration property to yes, this means authorization checking does not occur.

Example

```
exec sgw_addrpctogrp TGROUP1, SYV2, user
go
```

This isql example adds a standard RPC named "SYV2" to the transaction group named "TGROUP1." The user's alternate transaction processing region ID (HOST_LOGIN and HOST_PWD) is sent to the transaction processing region.

Deleting RPC names from a transaction group

To remove an RPC name from a transaction group:

```
exec sgw_droprpcfromgrp tran_group, rpc_name
```

where:

- *tran_group* is the name of the transaction group from which you want to delete the RPC.
- *rpc_name* is the name of the RPC you want to delete.

Example

To make sure the RPC named "SYV2" is no longer part of the "TGROUP1" transaction group:

```
exec sgw_droprpcfromgrp TGROUP1, SYV2
go
```

Modifying a transaction group

To change values in an existing transaction group:

```
exec sgw_modtrngrp tran_group, GROUP_LOGIN,
  GROUP_PWD, langrpc, langpwdlevel
```

where:

- *tran_group* is the name of the transaction group.
- *GROUP_LOGIN* is the alternate transaction processing region login that member transactions can use. When langpwdlevel is set to group, the GROUP_LOGIN overrides the HOST_LOGIN of the client calling this procedure. This value must be in uppercase.
- *GROUP_PWD* is the alternate transaction processing region password that member transactions can use. When langpwdlevel is set to group, this password overrides the HOST_PWD of the client calling this procedure. This value must be in uppercase.
- *langrpc* is the RPC name used to process SQL language requests. This is the name assigned to all language requests by users who use this transaction group.

- *langpwdlevel* is the source of the transaction processing region login information for language RPCs. It indicates whether the transaction processing region login ID and password should be passed to the transaction processing region with this transaction request, and if so, whether the user's HOST_LOGIN or the transaction group's GROUP_LOGIN information should be used. This parameter can be one of the following values:
  - none – do not send login information to the transaction processing region.
  - user – send the HOST_LOGIN and HOST_PWD.
  - group – send the GROUP_LOGIN and GROUP_PWD defined here.

Example

If the "TGROUP1" transaction group langpwdlevel is currently set to user, this isql example sets it to group:

```
exec sgw_modtrngrp TGROUP1, JOE, MOE, AMD2, group
go
```

The GROUP_LOGIN and GROUP_PWD are now set to "JOE" and "MOE," respectively. The language RPC remains AMD2, and the langpwdlevel is now group. If langpwdlevel is the only parameter value you are changing, enter:

```
exec sgw_modtrnprp TGROUP1,,,,group
go
```

The commas serve as placeholders for the unchanged parameters.

Deleting a transaction group

To delete a transaction group from the TRS security system:

```
exec sgw_droptrngrp tran_group
```

where *tran_group* is the name of transaction group you want to delete.

Example

This procedure deletes the transaction group named "TGROUP1":

```
exec sgw_droptrngrp TGROUP1
go
```

CHAPTER 6

Using Password Expiration Management (PEM) with TRS

Note This chapter applies only to LU 6.2.

What is PEM?

The Password Expiration Management (PEM) is a password management program that IBM provides with:

- CICS 3.3, through an optional PTF UN90057
- CICS versions 4.1 and later
- z/Series (formerly z/OS)

Sybase provides support for PEM as a feature of TRS for LU 6.2. This feature is not available for TRS connections to the mainframe using TCP/IP.

PEM server capabilities

The PEM server allows an APPC application to:

- Retrieve information regarding the success or failure of the host login process

- Validate any supplied user ID and password
- Determine when the host password expires
- Update the host password for a specified user ID

Starting a host transaction

When you attempt to start a host transaction from TRS, the request may fail due to a host security violation, such as an expired password or an incorrect password setup.

With PEM disabled, an LU 6.2 user cannot determine the exact cause of this security violation. SNA allows only a single error message to be returned to the error log, regardless of the cause.

With PEM enabled for TRS, if a host security violation occurs, TRS sends an error message to the client informing the user to execute a PEM RPC to obtain more information. The exact message depends on whether the request was made by an individual user ID or a transaction group's user ID. For example, TRS returns this error message if a security violation occurs as a result of a request made by an individual user ID:

```
34331, "The requested host transaction could not be
started because of a host security violation. Please
execute sgw_peminfopwd for more information."
```

Changing the host password

LU 6.2 TRS support for PEM also allows you to execute procedure calls to change the host password for either an individual user or for a transaction group at both the mainframe and TRS security levels.

PEM returns the following information in response to any of these procedure calls:

- The current successful host login date and time
- The last successful host login date and time
- The date and time the current host password expires (can be null if the password never expires)

- The revoke count (number of unsuccessful host logins since last successful logon)

Note PEM does not display the actual password itself.

The following sections explain how to implement and use PEM functionality as an additional feature of TRS for LU 6.2.

Implementing PEM functionality for LU 6.2 TRS

This section assumes that PEM is already installed and that all related host work is complete on the mainframe, as described in your IBM documentation.

CICS SIT table property

You may need to ask your CICS system programmer and the external security manager to change the setting of the CICS SIT table property, ISRDELAY=n. This property defines the intersystem refresh delay, which determines how long users remain signed on to the host when running transactions with the Inter System Communication (ISC) setting. Its setting may 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; for CICS version 4.1, this parameter is USRDELAY=0.

To implement TRS support for PEM after you install the TRS software, set the TRS PEMDest configuration property, which specifies the remote LU name (the name of the transaction processing region) in which the PEM server sign-on transaction resides on the host. See “PEMDest” on page 27 for more information.

Obtaining information about passwords

Use one of the following RPCs to obtain information about recent attempts to log in to the host and to determine the expiration date of a host password:

- sgw_peminfopwd – retrieves information about an individual user's host password expiration date and login attempts.
- sgw_peminfogrppwd – retrieves information about a transaction group's host password expiration date and login attempts.

The following sections describe syntax and usage notes for each procedure call.

User password information

To obtain information about an individual user's host password expiration date and recent login attempts, execute this RPC:

exec sgw_peminfopwd [*hostuserid, hostpwd*]

- If you include the *hostuserid* and *hostpwd* parameters, TRS passes the specified user ID and password to the PEM server.
- If you do not specify any parameters, TRS assumes you are requesting information about the client from which you are making the request. It passes one of the following to the PEM server, depending on whether security is enabled:
  - If TRS security is enabled (Security=yes), the client's *HOST_LOGIN* and *HOST_PWD*, as defined by sgw_addlog or by a previous sgw_pemchpwd procedure call
  - If security is not enabled (Security=no), the user ID and password that the client used to log in to TRS

Maximum length for the user ID and password is 8 characters each.

Group password

To obtain information about a transaction group's host password expiration date and recent logon attempts, execute this RPC:

exec sgw_peminfogrppwd *tran_group*

The *tran_group* parameter is required. It specifies the name of the transaction group for which you want logon and password information. TRS passes to the PEM server the transaction group's *GROUP_LOGIN* and *GROUP_PWD*, as defined by sgw_addtrngrp or by a previous sgw_pemchgrppwd procedure call.

Note You must have Gateway Control Access permission to execute this procedure call.

Changing passwords

With PEM enabled, you can use one of these RPCs to change a user's or group's host password:

- sgw_pemchpwd – change an individual user's host password.
- sgw_pemchgrppwd – change a transaction group's host password.

Note You must have Gateway Control Access permission to execute the procedure call for group password changes.

If you successfully change the password, this message appears on the client:

```
The password for host userid 'username' has been
successfully changed.
```

Syntax and usage notes for each procedure call are described in the following sections.

Changing an individual password

TRS clients can change their own host password by executing this RPC, where *newpwd* is the new host password for the client:

exec sgw_pemchpwd *newpwd, newpwd*

You must be logged in as the user whose password you want to change. Depending on whether security is enabled, TRS passes one of the following to the PEM server:

- The client's *HOST_LOGIN* and *HOST_PWD*, as defined by sgw_addlog or by a previous sgw_pemchpwd call, if security is enabled (Security=yes)
- The user ID and password that the client used to log on to TRS, if security is off (Security=no)

The user ID, current password, and new password must be up to 8 characters. You must enter the new password twice, as shown in the preceding syntax example.

This operation updates a user's host password at the mainframe security system, and, when TRS security is enabled, it also updates the user's *HOST_PWD* at the TRS security level.

Only the individual user can change his or her password on the host; the TRS administrator cannot perform this task.

Note When security is not enabled, changing the host password does not change the password under which are currently logged in. When you change your host password, you cannot execute any RPCs until you log out of TRS and log in with the correct password.

Changing a group's password

To change a transaction group's host password, you must have Gateway Control Access permission. Execute the following RPC:

exec sgw_pemchgrppwd *tran_group newpwd*, *newpwd*

The *tran_group* parameter is required. It specifies the name of the transaction group for which you want to change the password.

You must enter the new password (*newpwd*) twice, as shown.

This operation updates the host password of the group user ID at the mainframe security system, as well as the transaction group's *GROUP_PWD* at the TRS security level, which was last defined by sgw_addtrngrp or by a previous sgw_pemchgrppwd procedure call.

Setting up new users

You can use PEM procedure calls to access login information or change a user's host password, only if the user already has a valid host password that is known to the mainframe security system.

Note You cannot use the sgw_pemchpwd or sgw_pemchgrppwd procedure calls to set up the initial host password for a new user.

The TRS administrator coordinates host security setup for new users with the mainframe external security administrator:

- The TRS administrator uses the sgw_addlog or sgw_addtrngrp procedure to set up an individual or group user ID and initial host password at the TRS security level.
- The mainframe external security administrator implements the assigned user ID and host password at the mainframe security level.

After initial setup is complete, the new user should log in to the system and change the administrator-assigned password to a private one using the sgw_pemchpwd procedure call.

For more information about setting up new users, see "Adding a login" about using sgw_addlog.

For more information about setting up new transaction groups, see "Adding a transaction group" in Chapter 4 about using sgw_addtrngrp.

CHAPTER 7

Controlling a TRS

Note If you are enforcing security at TRS, the procedures described in this chapter require administration permissions. See “Adding a login” on page 105 for information about TRS administration permissions.

Controlling connections (LU 6.2 only)

If you are using LU 6.2, this section describes how to start a single connection and all connections, how to prevent inactive connections, and how to stop a connection gradually and abruptly.

Activating connections

To reactivate a single connection or restart all inactive connections, use one of the procedures described in this section.

By default, connections are active as a result of defining them. Connections may require reactivation if they have been made inactive, either as a result of using the sgw_deactcon RPC or a problem on the SNA network while the sgw_deactcon property is set to yes.

Activating a single connection

To activate a single connection:

```
exec sgw_actcon "con_number"
```

where *"con_number"* is the number of the connection you intend to start. This is the *connect number* with the value displayed in the sgw_status connections procedure. Enclose numeric parameter values in quotation marks.

Example

To activate connection number "1":

```
exec sgw_actcon "1"
go
```

Restarting all connections

To restart all connections:

```
exec sgw_actcon all
```

where the all option activates all connections, allowing you to recover when your SNA support stops or connections become inactive for any reason.

Marking connections as inactive

To have TRS mark a connection as "inactive" if it receives an unrecoverable error when trying to use the connection, set DeactCon=yes. When the error that caused the connection to be marked "inactive" is corrected, reactivate the connection.

Preventing inactive connections

To prevent TRS from marking connections "inactive," you can set the TRS DeactCon=non. Sybase recommends this option for remote sites that run unattended.

Deactivating a connection

To deactivate a connection, use either one of these procedures:

```
exec sgw_deactcon "con_number"
```

or

```
exec sgw_deactcon "con_number", force
```

where:

- *"con_number"* is the number of the connection you intend to deactivate that is displayed in the sgw_status connection procedure. Enclose numeric parameter values in quotation marks.
- force is optional. If you use the force option, the connection you specify ends, even if it is currently executing. However, on some TRS platforms, even a forced deactivate allows the current request to complete before deactivating the connection.

  If you do not use the force option, TRS allows any transactions in progress to complete before it deactivates the connection. While these transactions finish processing, the connection is considered to be "draining."

Example

To deactivate connection number "1":

```
exec sgw_deactcon "1", force
go
```

The force option causes connection number "1" to deactivate even if it is currently executing.

Deactivating LU 6.2 connections

This section describes how to deactivate connections in an LU 6.2 environment before you disconnect clients.

If you need to disconnect a client that is waiting for transaction results, you or your system programmer can use one of these methods to deactivate the connection before you disconnect the client:

- Using VTAM:

  ```
  VARY NET,INACT,ID=lu_name,FORCE
  ```

- Using isql:

  ```
  exec sgw_deactcon "con_number", force
  ```

Controlling regions (TCP/IP Only)

This section describes how to activate a single region or all regions in a TCP/IP environment and how to deactivate a region.

Activating a region

To activate a single region or to restart all inactive regions, use one of the following procedures.

Restarting all regions

To restart all regions:

```
exec sgw_actregion all
```

where the all option activates all regions, allowing you to recover when your TCP/IP support stops or regions become inactive for any reason.

Activating a single region

To activate a single region:

```
exec sgw_actregion region
```

where *region* is the name of the region you intend to activate.This is the name you assigned to the region in the sgw_addregion procedure.

Example

To activate the region named “TESTREG”:

```
exec sgw_actregion TESTREG
go
```

Deactivating a region

Deactivating a region prevents users from using that region.
To deactivate a region:

```
exec sgw_deactregion region
```

where *region* is the name of the region you intend to deactivate. This is the name you assigned to the region in the sgw_addregion procedure.

Example

To deactivate the region named “TESTREG”:

```
exec sgw_deactregion TESTREG
go
```

Disconnecting a client

You can force a particular client to disconnect. Generally, you use this command when you want to disconnect idle clients or clients having network problems. To disconnect a client:

```
exec sgw_disclient "client_number"
```

where *"client_number"* is the number of the client you intend to disconnect. Obtain the client number from the sgw_status clients procedure. Enclose numeric parameter values in quotation marks.

If you disconnect a client that invoked a long running transaction before the transaction ends, TRS deallocates the conversation and disconnects the client.

Example

To disconnect client number "7":

```
exec sgw_disclient "7"
go
```

If a transaction is in process, this command disconnects clients that are actively reading and processing results.

Controlling RPCs

You can take an RPC out of service by declaring it inactive. TRS rejects any client call to an inactive RPC name. A typical reason to deactivate an RPC is when the associated mainframe transaction is temporarily off line.

Activating an RPC

To make a defined RPC available (activate):

```
exec sgw_actrpc rpc_name
```

where *rpc_name* is the name of the RPC you intend to activate.

Example

To activate the SYV2 RPC:

```
exec sgw_actrpc SYV2
go
```

Deactivating an RPC

To make a defined RPC unavailable (deactivate):

```
exec sgw_deactrpc rpc_name
```

where *rpc_name* is the name of the RPC.

Example

To deactivate the SYV2 RPC:

```
exec sgw_deactrpc SYV2
go
```

Controlling tracing

The TRS tracing facility provides *entry/exit* tracing, tracing of the TRS interface with the back-end transport protocol, and TDS header and data tracing. When you enable tracing, tracing information is written to a set of error logs.

Ordinarily, you do not need to trace TRS activity. The tracing facility is provided to help Sybase Technical Support assist you if you call about certain errors. Tracing can also be useful for diagnosing local area network (LAN) and client application problems. For more information about tracing, see the sections describing the TRS TraceTRS, TraceProtocol, ProtocolTraceFile, and TDSTraceFile configuration properties in Chapter 2, “Creating a TRS Library.”

Mainframe-based tracing is described in the Mainframe Connect Server Option *Programmers Reference* guides. COBOL and PL/1 versions of this guide are available.

To enable or disable *entry/exit* tracing, set the server’s TraceEntryExit property. TRS entry/exit tracing accesses the following file:

- For UNIX:

  ```
  /<install_dir>/DC-15_0/srvname/log/srvname.trc
  ```

- For Windows:

  ```
  C:\<install_dir>\DC-15_0\srvname/log\srvname.trc
  ```

To enable or disable TDS tracing before TRS starts running, set the appropriate properties in the TRS configuration file. See the sections describing the TRS TraceTRS and TDSTraceFile configuration properties in Chapter 2, “Creating a TRS Library”.

To enable or disable protocol tracing before TRS starts running, set the appropriate properties in the TRS configuration file. See the sections describing the TRS TraceProtocol and ProtocolTraceFile configuration properties in Chapter 2, “Creating a TRS Library.”

You can also use DirectConnect Manager to enable and disable tracing while TRS runs.

Starting tracing

To start *TDS* tracing:

```
exec sgw_starttrace TDS
```

where *TDS* activates TDS tracing. TDS tracing accesses the following directory:

```
/<install_dir>/DC-15_0/srvname/log
```

To start *protocol* tracing:

```
exec sgw_starttrace PROT
```

where *PROT* activates tracing of the DirectConnect interface with the backend transport protocol layer, for either TCP/IP or LU 6.2.

- On Windows, back-end TCP/IP tracing goes into:

```
C:\<install_dir>\DC-15_0\srvname\log\trstcp.ngtcp
```

- On UNIX, back-end TCP/IP tracing goes into:

```
/<install_dir>/DC-15_0/srvname/log/ngtcp.trstcp
```

- On Windows, back-end LU 6.2 tracing goes into:

```
C:\<install_dir>\DC-15_0\srvname\log\trslu62.nglu62
```

- On UNIX, back-end LU 6.2 tracing goes into:

```
/<install_dir>/DC-15_0/srvname/log/nglu62.trslu62
```

If no parameter is entered, the default is *TDS*.

Stopping tracing

To stop TDS tracing:

```
exec sgw_stoptrace TDS
```

where *TDS* tracing is disabled.

To stop protocol tracing, use the following procedure:

```
exec sgw_stoptrace PROT
```

where *PROT* tracing is disabled for either LU 6.2 or TCP/IP.

If no parameter is entered, the default is *TDS*.

Controlling accounting

TRS allows you to record accounting information. This section describes how to record accounting at TRS. Mainframe-based accounting is explained in the Mainframe Connect Server Option documentation.

The TRS accounting facility records the following information:

- The name by which TRS is known.
- The RPC the named client calls.
- The connection name or mainframe name relevant to this RPC.
- The date and time that TRS sent the request to the transaction processing region.
- The time elapsed since the request was sent.

  The time elapsed count starts when TRS receives the request and continues until the final result row is sent to the client. Some applications, such as Data Workbench, read a few rows at a time, and then request more rows as the user requests them to be displayed, allowing the user to read the results. This time is included in the total duration.
- The total number of bytes sent and received with this RPC.
- The total number of Sybase TDS packets sent and received with this RPC. A packet is 512 bytes or less.

To turn on accounting before TRS starts running, set the appropriate properties in the TRS configuration file. See the sections describing the Accounting and AccountFile configuration properties in Chapter 3, “Configuring a TRS.”

Note You can also use DirectConnect Manager to enable and disable accounting while TRS is running.

Activating and deactivating accounting

To start and stop the TRS accounting facility while TRS is running:

```
exec sgw_startact
exec sgw_stopact
```

Executing these procedures is equivalent to setting Accounting=no.

Reading the accounting log

When you activate accounting, TRS writes the accounting records to the accounting log. See the AccountFile configuration property for the name of the accounting log file.

To display the accounting log:

```
exec sgw_dspact
```

Each accounting log record is returned in a row.

Stopping TRS

Generally, TRS runs continuously. If you need to deactivate TRS, use this procedure to disconnect each client and allow conversations in progress to finish first:

```
exec sgw_shutdown
```

Use this procedure to disconnect each client immediately without waiting for conversations in progress to finish:

```
exec sgw_shutdown now
```

With the preceding procedure, TRS does not accept any new client requests.

When you are ready to start TRS again, set the configuration properties described in Chapter 3, “Configuring a TRS.”

CHAPTER 8

Monitoring a TRS

Monitoring the status of TRS

You can use the sgw_status procedure to query the status of TRS. The status procedures show:

- The clients logged in to TRS
- The system on which they are running
- Remote procedure calls (RPCs) they call
- Connections they use
- Accounting information

The following command queries the status of TRS:

```
exec sgw_status options
```

The following are values for *options* in the sgw_status procedure, which is described in this chapter:

- clients
- connections (LU 6.2 only)
- regions (TCP/IP only)

- rpc
- parameters
- trace
- sum

Monitoring clients

To query the status of clients:

```
exec sgw_status clients
```

Table 8-1 shows the information this procedure displays for all active clients:

Table 8-1: Description of sgw_status clients results

Field	Description
Login	The login name of the user.
Client_Number	TRS issues a unique client number each time a user logs in. A user logged in more than once has the same *Login* and a different *Client_Number* for each connection to TRS. **Note** For LU 6.2, this number identifies the user's logins in the connections status display.
RPC_Name	The RPC called by the client: • If the request is a direct call from an Open Client DB-Library application, this field contains the RPC name specified in the dbrpcinit statement. • If the request is an indirect call from a Adaptive Server stored procedure, this is the RPC name that the stored procedure used when it called TRS.
Host_Tran	The name of the mainframe (host) transaction being invoked. This is the mainframe transaction associated with the RPC name. If a transaction is not in progress, this field is blank.
Client_Machine	The name of the machine on which the client program is running.
Con_Number (LU 6.2 only)	The connection number. TRS uses this number to represent the client's current SNA connection. If the client is not using any connections, this field is blank.

Field	Description
State	The state of the transaction. Valid values for this 2-character field are: • AL (allocation) – TRS is allocating a conversation (LU 6.2 only) to the displayed mainframe (host) transaction or opening a socket (TCP/IP only) to the mainframe transaction. • CQ (connection queue) – TRS (LU 6.2 only) is waiting for an available connection to the mainframe. • GC (TRS administration) – the user is executing TRS administration procedures or is using the Gateway Control Program. • ID (idle) – the client is connected to TRS, but a transaction is not in progress. • IT (idle in transaction) – TRS is invoking a long-running transaction. The client is between procedure calls but a conversation is active. • RS (reading server – TRS is waiting for the mainframe transaction to return results. • SH (site handler) – the client is Adaptive Server. • WA (waiting) – the conversation is allocating, and TRS is waiting for the first set of results. • WC (writing to client) – TRS is writing data to the client program.
Count	The number of buffers being written to the client program as part of the result set of the current transaction. Each buffer contains approximately 512 bytes of data.
Time	The length of time, in seconds, that the transaction has been running.
SPID	TRS uses this number internally.

You may find an entry in this screen with SH listed under the State field but without an associated Login field. This entry represents the site handler for a remote Adaptive Server.

To display the name of the remote server in the Login field, add this configuration command at Adaptive Server, and then restart Adaptive Server:

```
sp_addserver servers/srvname, local
```

Replace *srvname* with the name of the remote Adaptive Server.

Monitoring connections (LU 6.2 only)

To query the status of the connections that TRS uses:

```
exec sgw_status connections
```

This procedure displays the defined connections currently known to TRS and the status of each.

Table 8-2 shows the connection information that returns:

Table 8-2: Description of sgw_status connections results

Field	Description
Con_Number	This number represents the connection being described.
Status	The connection availability, which indicates whether the connection is currently available for use. Valid values are: • A (active) – the connection is available. • I (inactive) – the connection is not available. • D (draining) – the connection is not available. A connection is considered to be "draining" if you deactivated it while it was in use. It remains in draining status until the request completes, then becomes inactive.
Connection	The name of the SNA connection as defined in the sgw_addcon procedure.
Mode	The name of the mode used with this connection as defined in the sgw_addcon procedure.
Destsys	The name of the transaction processing region accessed by the connection as defined in the sgw_addcon procedure.
Host_Tran	The name of the mainframe transaction being invoked as defined in the sgw_addrpc procedure. If a transaction is not being invoked, this field value is Null.
Client_Number	The client currently using this connection. This is the same number used to identify the login of each client in the result of the sgw_status clients command. If the client is not using a connection, this field is blank.

Monitoring regions (TCP/IP only)

To query the status of the regions that TRS uses:

```
exec sgw_status region
```

Table 8-3 displays status information that is displayed:.

Table 8-3: Description of sgw_status region results

Field	Description
Region	The name of the transaction processing region as specified in sgw_addregion.
Host Name	The TCP/IP network host name as specified in sgw_addregion.
Port	The number of the port as specified in sgw_addregion.
Status	The region availability, which indicates whether the named region is currently available for use. Valid values: • A (active) – the region is available for use. • I (inactive) – the region is unavailable for use.

Monitoring RPCs

To see if an RPC is defined, as well as the transaction processing region (destination subsystem) it is associated with, use this command. (To see the RPCs in use, use the sgw_status clients procedure.) This statement displays the defined RPCs in TRS:

```
exec sgw_status rpc
```

Table 8-4 shows the information displayed:

Table 8-4: Description of sgw_status rpc results

Field	Description
RPC	The name of the RPC being called.
Status	The availability of the RPC, which indicates whether the named RPC is currently available for use. Valid values are: • A (active – the RPC is available for use. • I (inactive) – the RPC is unavailable for use.
Host_Tran	The name of the mainframe transaction to be invoked.
Security_Fields	The mainframe access permission requirements. This field specifies the administration procedure parameter values that TRS passes to the mainframe. Valid values are: • U (login ID) – the login ID is passed to the mainframe. • B (both) – both the ID and password are passed to the mainframe. • N (none) – login information is not passed to the mainframe.
Destsys	The name of the transaction processing region with which this RPC is associated as defined in the sgw_addrpc procedure.

Displaying TRS configuration properties

To display the current property settings in the TRS configuration file:

```
exec sgw_status parameters
```

Note See Chapter 3, “Configuring a TRS,” for complete information about setting up the TRS configuration file.

Table 8-5 describes the results:

Table 8-5: Description of sgw_status parameters results

Field	Description
Version	The version/release level of the current TRS and the platform and operating system on which it is running.
Server name	The name of the DirectConnect server.
Protocol type	The network protocol used by this TRS, either LU 6.2 or TCP/IP.
National language	The default language for TRS. (This is also set at the mainframe in SYGWMCST, which is the global customization module.)

Field	Description
Char set	The default character set for TRS. (This is also set at the mainframe in SYGWMCST, which is the global customization module.)
Direct RPCs disabled	Indicates whether TRS will accept an RPC directly from a client or whether all RPCs must be indirect, that is, routed through Adaptive Server. Valid values are: • Yes – indirect routing is required. • No – indirect routing is not required; direct routing is permitted.
Max users	The maximum number of users allowed to use TRS at one time.
Max site handlers	The maximum number of site handlers allowed. A site handler controls the network connection to a remote server.
Truncate longvarchar	The truncation flag for DirectConnect for DB2 UDB. (This is also set at the mainframe in SYGWMCST, which is the global customization module.) This indicates whether the data in fields of the datatype long varchar are to be truncated to 255 bytes and passed to the client. If the truncation flag is not used, varchar data is sent as text and image datatypes for 4.*x* TDS clients, or as long varchar datatype for 5.0 TDS clients. Valid values are: • Yes – the data is truncated. • No – other datatypes are returned.
Security enforced	This indicates whether TRS security is enabled or overridden. Valid values are: • Yes – security is enforced at TRS. • No – security is not enforced at TRS, except for the "sa" account. (The RPC security definition is sent to the mainframe, if specified in the RPC definition.)
Interfaces file	The complete path and file name of the *interfaces* file for this TRS.
RPC file	The complete path and file name of the file that contains RPC information for this TRS.
Security grp file	The complete path and file name of the file that contains security information for this TRS.
Accounting	Indicates if accounting is activated for this TRS. Valid values: • Yes – accounting is on. • No – accounting is off.
Connection file (LU 6.2 only)	The complete path name of the file that contains connection information for this TRS.

Field	Description
Con wait q (secs) (LU 6.2 only)	The number of seconds that the connection request waits in the queue for an available LU 6.2 connection.
Region file (TCP/IP only)	The complete path and file name of the file that contains region information for this TRS.

Requesting trace information

The TRS tracing facility provides TDS header and data tracing. When you enable tracing, TRS writes tracing information to the trace file name specified in the TDSTraceFile configuration property.

Usually, you do not trace TRS activity. The tracing facility is provided to help Sybase Technical Support understand what occurred if you have to call about specific errors.

To request information about the status of the trace facility:

```
exec sgw_status trace
```

Table 8-6 shows the information that the procedure displays:

Table 8-6: Description of sgw_status trace results

Field	Description
Version	The version and release level of the current TRS and the platform and operating system on which it is running.
Trace	The trace indicator, which indicates whether tracing is enabled for this TRS. Valid values are: • Active – the trace facility is enabled. • Inactive – the trace facility is disabled.
Logfile	The name of the file for this DirectConnect server where TRS log records are written.
TDSlog	The name of the file for this TRS that contains trace data between TRS and the mainframe.

Summary of clients in each listed state

To request the summary of all clients and their current state:

```
exec sgw_status summary
```

Table 8-7 lists shows how this command tabulates the number of clients in each state:

Table 8-7: Summary of clients and their current state

State	Count	Description
ID	0	Connected; no transactions
AL	0	Allocating conversation
WA	0	Waiting for first results
RS	0	Reading from host
WC	0	Writing to client
SH	0	Site handler
GC	1	Gateway control
CQ	0	Queued; awaiting connection
IT	0	Idle; transaction active

APPENDIX A

Sending Requests to TRS

Introduction

Clients using Sybase Mainframe Connect options (see page 7) can send requests to TRS to access mainframe data. TRS forwards the requests to the mainframe and returns results in the same format as the results that ASE returns. Communication between TRS and the mainframe is transparent to the client.

Description of request types

Clients can send two types of requests to TRS:

- SQL language requests
- Remote procedure calls (RPCs)

Requests can be sent to TRS two ways:

- *Direct* requests are RPCs or SQL language statements that access TRS without an intermediary server.
- *Indirect* requests invoke an RPC on Adaptive Server, ASE/CIS or Replication Server, which then sends a request to TRS.

If you are using the Mainframe Connect DB2 UDB Option, TRS directs requests to the AMD2 transaction at the mainframe. If you are using Mainframe Connect DB2 UDB for CICS and IMS, TRS directs requests to the SYRT transaction at the mainframe. (See "Configuring a default SQL language handler for TRS" on page 55.)

Clients can send SQL language requests and RPCs to the AMD2 transaction. TRS handles any request sent indirectly (that is, through Adaptive Server) as an RPC. Long-running transactions cannot be sent through Adaptive Server because Adaptive Server logs out of TRS after each request. This is not true for ASE/CIS.

Size of requests to AMD2

The AMD2 transaction can process language requests up to 32K. RPC parameters submitted to AMD2 must be *CHAR* parameters. The transaction concatenates multiple RPC parameters into one SQL statement for DB2 UDB. (Include sufficient blanks in each parameter to make a valid statement for DB2 UDB.)

Sending SQL statements to DB2 UDB

When sending SQL language requests to DB2 UDB, the client can send only SQL statements that are understood by DB2 UDB. See the Mainframe Connect DB2 UDB Option *Installation and Administration Guide* for CICS and IMS for information about SQL compatibility.

Accessing DB2 UDB data

Clients can send SQL statements that access DB2 UDB data to TRS directly or indirectly. TRS sends SQL language requests to the AMD2 mainframe transaction, which submits the SQL statements to DB2 using DB2's dynamic SQL facility. AMD2 performs the requested actions and returns results to TRS, which forwards them to the client.

Sending RPCs to TRS

As described in Chapter 3, "Configuring a TRS," TRS maps RPCs to mainframe transactions. If your site uses the Mainframe Connect DB2 UDB Option, RPCs are mapped either to AMD2 or to the Catalog RPCs, which retrieve specific catalog information about DB2 UDB system tables.

Clients can use any of the following methods to send an RPC to TRS:

- Using isql or another SQL utility, send the RPC using the execute command.
- Include the RPC in an Open Client application, and send the RPC directly to TRS.
- Send an RPC indirectly. A client can call a stored procedure in Adaptive Server that in turn sends an RPC to TRS. (Do not use this method for long-running transactions unless you are using the functionality of ASE/CIS.)

Sending RPCs directly to TRS

To send RPCs directly to TRS, use the execute command, which is described in this section. See the Sybase Adaptive Server Enterprise *Reference Manual: Commands* for detailed information about the execute command.

The syntax for the exec command is:

```
exec procedure_name [[@parameter_name=value]
 [,[@parameter_name=value...]]]
```

where:

- *procedure_name* is the name by which the RPC is defined to TRS.
- *@parameter_name=value* is the value assigned to one of the RPC parameters. Repeat this argument for each of the RPC parameters. The name is optional.

  @parameter_name=value allows you to enter the parameters in any order, as long as the mainframe program can recognize parameters by name. If you use this form for any parameter, you must use it for all parameters in the same exec statement.

Sending RPCs indirectly to the mainframe

To send an indirect request to the mainframe, a client application issues an RPC that resides on Adaptive Server.

After parsing and pre-processing the request, Adaptive Server sends the request and parameters to TRS for forwarding to a mainframe transaction. When the results return, they follow the same route in reverse.

If you set the TRS DirectPrevent configuration property to yes, TRS rejects all direct calls from client applications, requiring all requests to be sent indirectly. Routing all requests through Adaptive Server allows you to use additional front-end tools and provide additional security checks. Do not use this method if the client submits long-running transactions.

Stored procedures that call TRS

When Adaptive Server calls TRS, it follows the same procedure it does for any other call to a remote server. The stored procedure uses the exec statement to call the remote procedure and forward the parameters.

For an example of a stored procedure, see Chapter 3, “Configuring a TRS.”

Unsupported calls

This section lists the Sybase Open Client DB-Library and CT-Library calls that are not supported in this release.

DB-Library calls

Routines designed to process results of COMPUTE calls:

dbadata	dbaltlen	dbaltbind_ps
dbadlen	dbaltop	dbanullbind
dbaltbind	dbalttype	dbbylist
dbaltcolid	dbaltutypex	

Browse mode routines:

dbequal	dbtsput	dbtabcount
dbfreequal	dbcolbrowse	dbtabname
dbtsnewval	dbcolsource	dbtabsource
dbtsnewlen	dbtabbrowse	

Registered procedure routines:

dbncreate	dbreghandle	dbregparam
dbnpdefine	dbreginit	dbregwatch
dbregdrop	dbregnowatch	dbregwatchlist
dbreglist	dbregparm	dbsetnotifs

Network routines:

dbrecvpassthru	dbsendpassthru

Bulk copy routines:

bcp_batch	bcp_columns	bcp_moretext
bcp_bind	bcp_control	bcp_sendrow
bcp_colfmt	bcp_done	BCP_SETL
bcp_collen	bcp_exec	
bcp_colptr	bcp_init	

Two-phase commit routines:

abort_xact	commit_xact	scan_xact
build_xact_string	open_commit	start_xact
close_commit	remove_xact	stat_xact

Routines that process options:

dbclopt	dbisopt	dbsetopt

Other disallowed routines:

dbchange	DBMORECMDS	dbreadpage
DBCMDROW	DBNUMORDERS	dbwritepage
DBCURCMD	DBOFFSET	dbuset
dbgetoff	dbordercol	

Client-Library calls

Routines designed to process results of COMPUTE calls:

ct_compute_info

Browse mode routines:

ct_br_column	ct_br_table

Network routines:

ct_sendpassthru	ct_recvpassthru

Bulk copy routines:

blk_alloc	blk_drop	blk_rowxfer
blk_bind	blk_getrow	blk_sendrow
blk_colval	blk_gettxt	blk_sendtext
blk_default	blk_init	blk_srvinit
blk_describe	blk_rowalloc	blk_textxfer
blk_done	blk_rowdrop	

APPENDIX B

Testing a TRS Installation with Sample Programs

Note This appendix describes the steps required to define a sample connection or region and RPC *for testing only*. The administration procedures for defining regions and RPCs to TRS are described in detail in Chapter 3, "Configuring a TRS." Also, see the "Configuration quick-start" on page 45.

When to test your installation

Use the instructions in this appendix after all of the mainframe access product components are installed at the workstation and at the mainframe.

Where to find the sample programs

Sample programs are located in these directories:

- For Windows: *C:\<install_dir>\DC-15_0\sample\trs*
- For UNIX: */<install_dir>/DC-15_0/sample/trs*

Note The samples require Open Client DB-Library on the workstation.

How to test your TRS installation

Follow the steps in this section to ensure that TRS is installed correctly. This section describes how to define a single region and RPC, and how to test them before you define other regions.

Starting TRS

Start DirectConnect for z/OS Option with TRS enabled. Run the samples with security disabled by setting the TRS Security=no.

Defining the connection for Windows (LU 6.2 only)

❖ **To define the test connection**

1 Log in to TRS as "sa" using isql or your preferred dynamic SQL utility, for example:

```
isql -Sservice_name -Usa -P
```

where *service_name* is the unique name of this TRS.

2 At the prompt, enter a command similar to the following, replacing the parameter values shown here with values that are appropriate for your installation.

```
exec sgw_addcon con_name, region, mode,
"max_sessions"
```

where:

- *con_name* is the name assigned to this connection and the name by which the connection is known to your SNA support. For different platforms, this parameter corresponds to different values. See the Mainframe Connect DirectConnect for z/OS Option *Installation Guide* for specific information about connection name parameter values.
- *region* specifies the remote LU name of the mainframe transaction processing region in this parameter. This is the Virtual Telecommunications Access Method (VTAM) APPLID name to which this connection is bound. An entry in this field is required.

All RPCs that use this connection to access the mainframe must have this same value specified as the *region* in their RPC definitions. (See "Adding an RPC" on page 52.)

- *mode* needs to match this value to the name of the mode defined to the mainframe and to the local SNA support for this connection (up to 8 characters). For different platforms, this parameter corresponds to different values. See the Mainframe Connect DirectConnect for z/OS Option *Installation Guide* for specific information about the mode name parameter value.
- *"max_sessions"* is the maximum number of sessions that can run concurrently over this connection. If you use parallel sessions, enter a value between 2 and 254. If you do not use parallel sessions, this value can only be 1. Enclose numeric parameter values in quotation marks.

Note Check with your SNA System Administrator to make sure this number is not larger than the maximum number of sessions (for this mode) defined to the SNA subsystem.

Example

In this isql example, "SYBLU01" is the connection name, "CICSQA" is the region name, "SYBMODE" is the mode name, and "1" is the number of maximum sessions:

```
exec sgw_addcon SYBLU01, CICSQA, SYBMODE, "1"
go
```

Defining the test region (TCP/IP only)

❖ **To define the test region**

1 Log in to TRS as "sa" using isql or your preferred dynamic SQL utility:

```
isql -Sservice_name -Usa -P
```

where *service_name* is the unique name of this TRS.

2 At the isql prompt, enter a command similar to the following, replacing the parameter values shown here with values that are appropriate for your installation.

```
exec sgw_addregion region, hostname, portnumber,
regiontype
```

where:

- *region* is the value used within TRS only. The value you specify here must match the value you specify in the region parameter of the sgw_addrpc procedure. This name can be up to eight characters.
- *hostname* is the value you specify for the TCP/IP network host name. This is the name corresponding to the mainframe in your */etc/hosts* file or in your NIS map. This name can be up to 30 characters.
- *portnumber* is the number you specify that must match the port number on which the CSKL transaction listens. It can be any number between 1024 and 9996. (This is not the same as the port number used to configure the *interfaces* file.)
- *regiontype* is the type of the mainframe processing environment specified by the region parameter. Valid values are CICS, MVS, and IMS. If you do not specify a value, the region type defaults to CICS.

Example

In this example, “CICSQA” is the region, “BLUES” is the host name, “3003” is the port number that the CICS Listener transaction is running on, and “CICS” is the region type:

```
exec sgw_addregion CICSQA, BLUES, "3003", CICS
```

Defining the test RPC

Define an RPC to execute in the specified region. The SYM2 transaction is a simple CICS transaction that fabricates data and does not require external resources such as DB2 UDB.

- At the prompt, enter a command similar to the one shown below.

```
exec sgw_addrpc rpc_name, tran_id, region,
 security
```

where:

- *rpc_name* is the name of the remote procedure that the client uses to call this RPC. The name can be up to 30 characters.
- *tran_id* is the name of the associated mainframe transaction that is called when a client requests the named procedure. The value of this field must be in uppercase. For CICS, use 4 characters. For IMS, use up to 8 characters.

- *region* (*LU 6.2 only*) specifies the remote LU name of the region in this parameter. Set this value to match the VTAM APPLID of the CICS or IMS region (the destination subsystem) in which the transaction (specified in tran_id) executes.

  At least one defined connection must have this value specified as its region. See also "Adding a connection configuration" on page 48. An entry in this field is required.

- *region* (*TCP/IP only*) is used within TRS only to represent the CICS region name. It must match the value you specify for the region parameter in the sgw_addregion procedure. See "Defining regions to TRS" on page 50. An entry in this field is required.

- *security* specifies the type of user login information to be passed to the transaction processing region:

  - Using LU 6.2, the information is passed in the conversation-level security fields of the SNA LU 6.2 Function Management Header 5 (FMH-5).
  - Using TCP/IP, these fields are sent to the CICS Listener Transaction when the CICS transaction is started.

  The security parameter can have any of the following values to specify which information is sent:

  - none – do not send login information to the mainframe.
  - userid – send only the user ID to the mainframe.
  - both – send both the user ID and the password to the mainframe.

  For example, if you use native CICS security, the none value corresponds to the CICS security option NONE, userid corresponds to IDENTIFY, and both corresponds to the security option VERIFY.

Example

```
exec sgw_addrpc SYM2, SYM2, CICSQA, none
```

where:

- SYM2 (first entry) is the RPC name.
- SYM2 (second entry) is the transaction ID at the mainframe.
- *CICSQA* is the CICS region name.
- none indicates that user IDs are not passed to the mainframe.

The CICS region name (CICSQA in the preceding example) must match the following:

- For TCP/IP, the *region* name given in the sgw_addregion procedure
- For LU 6.2, the *region* parameter in the sgw_addcon procedure

Running the sample

At the dynamic SQL utility prompt, enter the following to run the SYM2 sample:

```
exec SYM2 a, 4
```

The output should look like this:

```
TESTDATA

 ------------------------------------------------
          U6T42P01
 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

          U6T42P01
 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

          U6T42P01
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

          U6T42P01
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

 (4 rows affected, return status = 0)
```

Checking for error messages

The TRS request can return any of several types of error messages. Some messages are written to the error log at TRS, and others are returned to the client.

Looking at additional sample programs

After you successfully run the SYM2 sample, continue with some of the other samples provided in the following directories and in the *README* file in that directory:

- For Windows: *C:\<install_dir>\DC-15_0\sample\TRS\sym2*
- For UNIX: */<install_dir>/DC-15_0/sample/TRS/sym2*

Define the samples to TRS using the administration procedures described previously.

SYVn transactions

The SYVn transactions read a VSAM file and return the records.
The SYVn RPC passes two parameters: a starting and an ending byte offset.

Add the RPC using the sgw_addrpc procedure described in "Defining the test RPC" on page 156. These transactions call the following programs:

- SYV1 calls the PL/I program, SYCASAV1.
- SYV2 calls a COBOL program, SYCASAV2.
- SYV3 calls an assembler program, SYCASAV3, which is supplied on the mainframe.

To execute the SYVn RPC you defined, do *one* of the following:

- Enter the following command at the dynamic SQL utility prompt. (Replace *SYVn* with SYV1, SYV2, or SYV3.)

```
exec SYVn 0,9999
```

- If the directory containing the samples is on the search path, enter this command at the server console prompt:

```
SYVn 0,9999
```

Output should look like this:

```
SYGWLSA1
 ------------------------------------------------
 sample vsam rpc data rec 0 sample vsam rpc data rec 0
 sample vsam rpc data rec 1 sample vsam rpc data rec 1
 sample vsam rpc data rec 2 sample vsam rpc data rec 2
 .
 .
 .
 (10 rows affected, return status = 0)
```

SYDn transactions

SYD*n* transactions execute a static DB2 UDB query to one of the DB2 UDB sample tables. The parameter is the department number. Add the RPC using the sgw_addrpc procedure described in "Defining the test RPC" on page 156.

To execute the SYDn RPC, do *one* of the following:

- Enter this command at the isql prompt (replace *SYDn* with SYD1 or SYD2):

```
exec SYDn D11
go
```

- If the directory containing the samples is on the search path, enter this command at the server console prompt:

```
SYDn D11
```

Output should look like this:

```
LAST_NAME EMP_DEPT EMP_PHONE SALARY
 --------- -------- --------- -------------
 ADAMSON     D11     4510       25,280.00
 BROWN       D11     4501       27,740.00
 PIANKA      D11     3782       22,250.00
 STERN       D11     6423       32,250.00
 WALKER      D11     2986       20,450.00
 LUTZ        D11     0672       29,840.00
 SCOUTTEN    D11     1682       21,340.00
 YOSHIMURA   D11     2890       24,680.00
 JONES       D11     0942       18,270.00

 (9 rows affected, return status = 0)
```

The rest of the sample transactions demonstrate the Open ServerConnect programming techniques.

Looking at catalog RPC scripts

If you plan to use the Catalog RPCs, see the appropriate section in this guide for instructions on running the *addcat* installation script.

You can find the scripts that install, test, and delete the Catalog RPCs in the following directories:

- For Windows: *C:\<install_dir>\DC-15_0\scripts*
- For UNIX: *<install_dir>/DC-15_0/scripts*

APPENDIX C

Localization

What is localization?

Localization is the process of setting up an application to run in a particular national language environment. A localized application:

- Generates messages in a local language and character set
- Uses local datetime formats

A *locale* name is a character string that represents a language, character set, and sort order combination. For example, the *locale* name "fr" might represent the following language, character set, and sort order combination:

```
french/iso_1/binary
```

Sybase predefines *locale* names, which are listed in the *locales* file.

How servers handle conversions

When a localized client application connects to TRS, Adaptive Server, or Open Server, the server checks to see if it supports the client's language and character set. If it does, then the server:

- Automatically handles all character set translation
- Issues server messages in the client's language and character set

If TRS does not support the language or character set sort of the client, it issues a warning message to this effect, and Client-Library fails the connection. However, DB-Library accepts the connection.

Table C-1 describes these client and server behaviors:

Table C-1: Localization translation behaviors

Does server support client character set?	Does server support client language?	ASE server behavior	Open Server behavior	Client-Library behavior	DB-Library behavior
yes	yes	Performs all necessary message translation and character set conversion	Performs all necessary message translation and character set conversion	Operates normally	Operates normally
no	yes	N/A for Adaptive Server, because when Adaptive Server supports a language, it supports all character sets for that language	Uses the language and character set of the Open Server application	N/A for Adaptive Server; fails the connection for Open Server	N/A for Adaptive Server; accepts the connection for Open Server
yes	no	Uses the language us_english and the client's character set	Uses the language and character set of the Open Server application	Fails the connection	Accepts the connection
no	no	Uses the language us_english and the character set ascii_7	Uses the language and character set of the Open Server application	Fails the connection	Accepts the connection

Environment variables for localization

TRS examines environment variables when determining which language, character set, sort order, and datetime formats to use for an application.

TRS uses standard POSIX localization environment variables.

Some systems automatically set environment variables when a user logs in. If your system does this, either reset the variables after logging in or make sure that their automatic values correspond to an entry in the Sybase *locales* file.

Table C-2 lists the environment variables that are related to TRS localization:

Table C-2: TRS localization environment variables

Environment variable	Definition	When
LC_ALL	Indicates which language and character set to use for messages, datatype conversions, and datetime formats.	• The application calls cs_ctx_alloc. • The application calls cs_locale with type as *CS_LC_ALL* and buffer as NULL.
LC_CTYPE	Indicates which character set to use for datatype conversions.	The application calls cs_locale with type as *CS_LC_CTYPE* and buffer as NULL.
LC_COLLATE	Indicates which collating sequence (sort order) to use when sorting and comparing character data.	The application calls cs_locale with type as *CS_LC_COLLATE* and buffer as NULL.
LC_MESSAGE	Indicates which language and character set to use for messages.	The application calls cs_locale with type as *CS_LC_MESSAGE* and buffer as NULL.
LC_TIME	Indicates which language to use when converting between datetime and character datatypes. LC_TIME controls the following: • Month names and abbreviations • Datepart ordering • Whether the "am/pm" string is used	The application calls cs_locale with type as *CS_LC_TIME* and buffer as NULL. When an application calls cs_locale, Client–Library examines *LANG* if the cs_locale buffer is NULL and the *LC_ALL* variable corresponding to type is not defined.

Environment variable	Definition	When
LANG	Indicates which language, character set, and sort order to use for messages, datatype conversions, and datetime formats. **Note** Open Client and Open Server products search for LANG if they cannot find *LC_ALL*.	The application calls ct_ctx_alloc, Client-Library examines LANG if *LC_ALL* is not defined. When an application calls cs_locale, Client–Library examines LANG if the cs_locale buffer is NULL and the *LC_ALL* variable corresponding to type is not defined.

Localization files

This section contains information on Sybase files that are related to localization.

Note The directories shown in this appendix are for a Windows platform. For UNIX platforms, the directory path is *<install_dir>/DC-15_0/connectivity/locales* or *<install_dir>/DC-15_0/connectivity/charsets*.

Where localization files come from

Open Client and Open Server products, including TRS, come with the files to support one language and one or more character sets and sort orders.

At installation time, these files are automatically loaded into the *C:\<install_dir>\DC-15_0\connectivity* directory tree, in the locations illustrated in Table C-3.

The files to support additional languages are packaged as “Language Modules for Connectivity.”

When you install a language module, the language, character set, and sort order files to support the new language are automatically loaded into the *C:\<install_dir>\DC-15_0\connectivity* directory tree in the correct locations.

Location of localization files

Two directories in the *C:\<install_dir>\DC-15_0\connectivity* directory tree contain files related to localization:

- *C:\<install_dir>\DC-15_0\connectivity\locales*, which contains the *locales* file (*locales.dat*) and a subdirectory for each available language.
- *C:\<install_dir>\DC-15_0\connectivity\charsets*, which contains a subdirectory for each available character set.

Table C-3 shows where localization files are located in the *C:\<install_dir>\DC-15_0\connectivity* directory tree:

Table C-3: Location of localization files in the C:\<install_dir>\DC-15_0\connectivity directory

Subdirectory			
charsets	*charset_name*	*binary.srt* *charset.loc* *dictionary.srt* *noaccents.srt* *nocase.srt* *nocasepref.srt*	
locales	*language_name*	*charset_name*	*blklib.loc* *ctlib.loc* *common.loc* *cslib.loc* *oslib.loc* *trslu62.loc* *trdtcp.loc* *trstcp.loc*
	locales.dat		

Table C-4 shows information about some localization files:

Table C-4: Files related to localization

File name	File location	What it contains
locales.dat	*C:\<install_dir>\DC-15_0\connectivity\locales*	Entries that map a locale name to a language and character set. This is the *locales* file. For more information, see “Locales file” on page 166.
common.loc	*C:\<install_dir>\DC-15_0\connectivity\locales\language_name\charset_name*	Common information for the *language_name* language and *charset_name* character set, including date names and orders, and money formats and symbols.

File name	File location	What it contains
charset.loc	*C:\<install_dir>\DC-15_0\connectivity\locales\ language_name\charset_name*	Character set information for the *language_name* language and *charset_name* character set.
binary.srt	*C:\<install_dir>\DC-15_0\connectivity charset\charset_name*	The binary sort order for the *charset_name* character set.

*.loc files

The *C:\<install_dir>\DC-15_0\connectivity\locales\language_name\charset_name*.loc* files contain product error messages in the language and character set specified by their parent directories.

These files allow TRS to report errors in a specific language and character set.

Character set files

The *C:\<install_dir>\DC-15_0\connectivity\charsets\charset_name** files contain information related to a particular character set, including sort order, case, and accent information.

Locales file

The *locales* file associates *locale* names with languages, character sets, and sort orders. TRS uses the *locales* file when loading localization information.

The *locales* file, called *locales.dat*, is located in the *C:\<install_dir>\DC-15_0\connectivity\locales* directory.

The *locales* file directs TRS to language, character set, and sort order names, but it does not contain actual localized messages or character set information.

Format of locales file entries

The *locales* file has platform-specific sections. An entry defines a locale as the combination of a language, character set, and sort order:

```
locale = locale_name, language_name, charset_name
[,sort_order_name]
```

If the sort order is not specified, it is “binary.”

When the locale being defined is the default for the site, the locale_name is “default.” For example, this entry defines the default *locale* as us_english with the iso_1 character set and binary sort order:

```
locale = default, us_english, iso_1
```

How Client-Library and Server-Library set up default localization values

When a Client-Library or Server-Library application calls the CS-Library routine cs_ctx_alloc to allocate a context structure, CS-Library loads default localization information into the new context structure.

To load default localization information, CS-Library follows these steps:

1 CS-Library looks for a *locale* name, searching for the LC_ALL and LANG. environment variables, in order:

   - If LC_ALL is defined, CS-Library uses its value as the *locale* name. If LC_ALL is not defined but LANG is defined, CS-Library uses its value as the *locale* name.
   - If neither LC_ALL nor LANG is defined, CS-Library uses a *locale* name of “default.”

2 CS-Library looks up the *locale* name in the *locales* file to determine which language and character set are associated with it.

3 CS-Library loads localized messages and character set information appropriate to the language and character set determined in step 2.

This process provides the new context structure with all of the localization information that it needs.

APPENDIX D

TRS Process User Exits

Introduction

This appendix describes the steps to implement and use the TRS process user exits for both LU 6.2 and TCP/IP. TRS allows you to create user exits that are invoked from the TRS application prior to executing the actual event.

Supported user exits

TRS supports user exits corresponding to Open Server-defined Connect and Disconnect events. Within user exits, you are able to manipulate user ID and user password information prior to the event being executed by TRS. It is not necessary to modify any of this information to implement user exits; however, you may need to manipulate the password for security reasons for an application. For example, a user's password may be modified to become a time-restricted password for interpretation by an authentication server.

These are the Open Server-defined events for which *user exits* are supported:

- Connect
- Disconnect

Note The directories shown in this appendix are for UNIX platforms. For Windows platforms, the variable *C:\<install_dir>\DC-15_0* is used.

Connect

The Connect user exit provides the ability to override both the user ID and user password. The user exit is called at the end of all DirectConnect connection processing but *prior to* the TRS connection processing. This allows the user ID and user password to remain intact for Open Client and Open Server connections and change for connections from TRS to DB2 UDB.

The TRS connection handler will query the length of the user exits returned user ID and password string buffers to determine data content. When data is present, it will be transferred to the TRS User ID and Password buffers.

Refer to this directory for an implementation example:

`/<install_dir>/DC-15_0/srvname/sample/trs/ue/ue_connect.cpp`

Disconnect

The Disconnect user exit is called only when the client disconnects from TRS. All parameters are passed as constants and cannot be modified.

Refer to this directory for an implementation example:

`/<install_dir>/DC-15_0/srvname/sample/trs/ue/ue_disconnect.cpp`

Implementing user exits

TRS user exits for Open Server Connect and Disconnect events should be written and tested using the sample exits and test harness provided at */<install_dir>/DC-15_0/srvname/sample/trs/ue*. Following is a description of the source, header, *makefiles*, libraries and executable files, many of which are dependent on your platform.

The source files:

- *ue_connect.cpp* – connection event sample user exit code.
- *ue_disconnect.cpp* – disconnect event sample user exit code.
- *ue_test.cpp* – test harness to invoke the Connect and Disconnect user exits.

The header files, depending on your platform:

- *ue_platform.h* – platform required header for implementing user exits where <platform> is one of: aix, hpux, sol, nt.
- *ue_classes.h* – required by *ue_test.cpp*.
- *ue_global.h* – required user exit definitions.

The *makefiles*, depending on your platform:

- *makeexe.platform*

  where *platform* is aix, hpux, sol, or nt.

  Generates *ue_test <platform>*, the *user exit* test harness.

- *makefile.platform*

  where *platform* is aix, hpux, sol, or nt.

  Generates required library *libtrsue_platform.ext*

  where *ext* is .so for AIX and Sun Solaris, .sl for HP-UX, and dll for Windows.

These libraries, depending on your platform:

- *libue_platform.a*

  Required library for testing user exits with *ue_test.platform*,

  where *platform* is: aix, hpux, sol, or nt.

- *libtrsue_platform.ext*

  Library containing user exits generated by *makefile.platform*

where:

- *platform* is aix, hpux, sol, or nt.
- *ext* is .so for AIX and Sun Solaris, .sl for HP-UX, and .dll for Windows

These executables, depending on your platform:

- *ue_test.platform*

where *platform* is aix, hpux, sol, or nt.

Test harness produced from *makeexe.platform*:

One parameter is required by *ue_test.platform* which specifies the dynamic library to be loaded, for example:

```
ue_test.<platform> ./libtrsue_<platform><ext>
```

where:

- *platform* is aix, hpux, sol, or nt
- *ext* is .so for AIX and Sun Solaris, .sl for HP-UX, and .dll for Windows.

Configuring TRS to implement user exits

To implement TRS user exits, use these properties:

- ProcessExitEnabled

  Set to yes to enable the use of user exits. Only the exits that you have defined and added to your user exit library will be invoked.

- ProcessExitFile

  Full path and name of the user exit-shared library that you have created. From the sample code, *libtrsue_platform.ext* is its equivalent.

- TraceProcessUserExits

  Traces entry/exit points of function call to each of the user exits that you have defined. Normal setting is no; however, a setting of yes will help you determine execution through your exits.

Testing user exits

To test the provided samples:

```
execute makefile.platform
```

followed by:

```
makeexe.platform
```

Continue by executing this:

```
ue_test.platform ./libtrsue platform.ext
```

For example, on Solaris this is required:

```
make -f makefile.sol

make -f makeexe.sol

ue_test.sol ./libtrsue.sol.so
```

Note Although exits other than Connect and Disconnect are provided in the sample */<install_dir>/DC-15_0/servers/srvname/sample/trs/ue*, these exits are not supported at this time.

Interface specifications

Following are the required interfaces and their descriptions for implementing the Connect and Disconnect user exits.

ue_connect

Description

Connects the event user exit defining what actions are to be performed prior to TRS connecting to DB2 UDB.

Syntax

```
TRS_RETCODE TRS_PUBLIC
ue_connect (status, serviceName, serviceNameLength, applicationName,
applicationNameLength, userId, pUserIdLength, password,
pPasswordLength, pOutUserId, pOutPwd)

  TRS_STATUS        status;
```

```
const char*     serviceName;
const int       serviceNameLength;
const char*     applicationName;
const int       applicationNameLength
const char*     userId;
int*            pUserIdLength;
const char*     password;
int*            pPasswordLength;
char**          pOutUserId;
char**          pOutPwd;
```

Table D-1: TRS_RETCODE values

Value	Description
eTRS_FAIL	Indicates failure within *ue_connect( )*
eTRS_SUCCEED	Indicates success within *ue_connect( )*
eTRS_NOTIMPLEMENTED	Indicates stubbed out implementation of *ue_connect( )*
eTRS_DATAMODIFIED	Indicates that a pointer variable has been modified and implies eTRS_SUCCEED

Parameters

status

The state of the Service Library invoking *ue_connect( )*. Table D-2 describes the legal value for status:

Table D-2: Legal status values

Value	Description
eGood_	Always used
eObjNotFound_	Reserved for future use
eObjNotValue_	Reserved for future use
eFatal_	Reserved for future use

serviceName

Name of the service to which the connection was made.

serviceNameLength

Length of the *serviceName*.

*applicationNam*e

Name of the application from which the connection was made.

applicationNameLength

Length of the *applicationName*.

userId
ID of the connecting user.

pUserIdLength
Pointer to length of *userId*.

password
Password associated with *userId.*

pPasswordLength
Pointer to length of *password*.

pOutUserId
A pointer to a character string that must be allocated by this routine and may contain a modified *userId*.

pOutPwd
A pointer to a character string that must be allocated by this routine and may contain a modified *password*.

Usage

See syntax.

Comments

- *ue_connect()* must return eTRS_DATAMODIFIED if either pOutUserId or pOutPwd has been allocated.
- pUserIdLength must be updated to reflect the length of pOutUserId when pOutUserId has been allocated.
- pPasswordLength must be updated to reflect the length of pOutPwd when pOutPwd has been allocated.
- malloc() should be used to allocate space for pOutUserId and pOutPwd

Warning! By granting control to this user exit, DirectConnect has temporarily forfeited the management of Open Server threads. The result is that DirectConnect cannot ensure against *ue_connect* monopolizing execution nor the ability of *ue_connect* to create a deadlock. Please take precautions to prevent this.

Example

See a sample implementation at this location:

/<install_dir>/DC-15_0/servers/sample/trs/ue/ue_connect.cpp

ue_disconnect

Description

Defines what actions are to be performed prior to a client disconnecting from TRS.

Syntax

TRS_RETCODE TRS_PUBLIC
ue_disconnect (status, serviceName, serviceNameLength, applicationName, applicationNameLength, userId, pUserIdLength)

```
TRS_STATUS      status;
const char*     serviceName;
const int       serviceNameLength;
const char*     applicationName;
const int       applicationNameLength
ccnst char*     userId;
int*            pUserIdLength;
```

Table D-3: TRS_RETCODE values

Value	Description
eTRS_FAIL	Indicates failure within *ue_disconnect( )*
eTRS_SUCCEED	Indicates success within *ue_disconnect( )*
eTRS_NOTIMPLEMENTED	Indicates stubbed out implementation of *ue_disconnect( )*

Parameters

status

The state of the Service Library invoking *ue_disconnect( )*. Table D-4 describes the legal value for status:

Table D-4: Legal status values

Value	Description
eGood_	Always used
eObjNotFound_	Reserved for future use
eObjNotValue_	Reserved f or future use
eFatal_	Reserved for future use

serviceName

Name of the service to which the connection was made.

serviceNameLength

Length of the *serviceName*.

*applicationNam*e

Name of the application from which the connection was made.

applicationNameLength
Length of the *applicationName*.

userId
ID of the connecting user.

pUserIdLength
Pointer to length of *userId*.

Usage

See syntax.

Comments

- *ue_disconnect* allows you to perform varying functions related to disconnects. Although *pUserIdLength* is defined as a pointer, its modification is meaningless with this release.

Warning! By granting control to this user exit, DirectConnect has temporarily forfeited the management of Open Server threads and DirectConnect cannot ensure against ue_disconnect monopolizing execution, nor ue_disconnect's ability to create a deadlock. Please use precautions to prevent this.

Example

See a sample implementation at the following location:

/<install_dir>/DC-15_0/servers/sample/trs/ue/ue_connect.cpp

Glossary

accept
Establishment of a SNA or TCP/IP connection between Mainframe Connect Server Option and Mainframe Connect DirectConnect for z/OS Option.

access service
The named set of properties, used with an access service library, to which clients connect. Each DirectConnect server can have multiple services.

access code
A number or binary code assigned to programs, documents, or folders that allows authorized users to access them.

access service library
A service library that provides access to non-Sybase data contained in a database management system or other type of repository. Each such repository is called a "target." Each access service library interacts with exactly one target and is named accordingly. See also **service library**.

ACSLIB
See **access service library**.

Adaptive Server 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.

Adaptive Server Enterprise/Component Integration Services
Includes a variation of ASE that provides a Transact-SQL interface to various sources of external data. Component Integration Services allows ASE to present a uniform view of enterprise data to client applications.

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 server management. See also **remote procedure call**, **service library**.

ADMLIB
See **administrative service library**.

Advanced Interactive Executive
The IBM implementation of the UNIX operating system. The RISC System/6000, among other workstations, runs the AIX operating system.

advanced program-to-program communication
Hardware and software that characterize the LU 6.2 architecture and its implementations in products. See also **logical unit 6.2**.

AIX See **Advanced Interactive Executive.**

AMD2 The component of the Mainframe Connect DB2 UDB Option that allows clients to submit SQL statements to DB2 UDB. It is a CICS transaction that receives SQL statements sent from Mainframe Connect DirectConnect for z/OS Option and submits them to DB2 UDB, using the DB2 UDB dynamic SQL facility. It also receives the results and messages from DB2 UDB and returns them to Mainframe Connect DirectConnect for z/OS Option.

American Standard Code for Information Interchange The standard code used for information interchange among data processing systems, data communication systems, and associated equipment. The code uses a coded character set consisting of 7-bit coded characters (including a parity check, 8 bits).

API See **application program interface**.

APPC See **advanced program-to-program communication**.

application program interface The programming language interface between the user and Mainframe Connect Client Option or Mainframe Connect Server Option. The API for Mainframe Connect Client Option is Client-Library. The API for Mainframe Connect Server Option is Gateway-Library.

ASCII See **American Standard Code for Information Interchange**.

ASE See **Adaptive Server Enterprise**.

ASE/CIS See **Adaptive Server Enterprise/Component Integration Services**.

batch A group of records or data processing jobs brought together for processing or transmission.

bind In the Sybase environment, this term has different meanings depending on the context:

- In CICS, it is an SNA command used to establish a connection between LUs, or a TCP/IP call that connects an application to a port on its system.
- In DB2 UDB, it compiles the Database Request Module, the precompiler product that contains SQL statements in the incoming request, and produces an access plan, a machine code version of the SQL statements that specifies the optimal access strategy for each statement.
- In the mainframe access product set, it establishes a connection between a TRS port and a CICS or IMS region.

bulk copy transfer A transfer method in which multiple rows of data are inserted into a table in the target database. Compare with **destination-template transfer** and **express transfer**.

call level interface A programming style that calls database functions directly from the top level of the code. Contrast with **embedded SQL.**

catalog A system table that contains information about objects in a database, such as tables, views, columns, and authorizations.

catalog RPC A component of the Mainframe Connect DB2 UDB Option that allows clients to access DB2 UDB system catalogs. It uses an interface compatible with the catalog interface for the ODBC API.

catalog stored procedure A procedure used in SQL generation and application development that provides information about tables, columns, and authorizations.

character set A set of specific (usually standardized) characters with an encoding scheme that uniquely defines each character. ASCII is a common character set.

CICS See **Customer Information Control System**.

CICS region The instance of CICS.

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 responsible for the user interface that sends requests to applications acting as servers. See also **client/server**.

Client-Library A library of routines that is part of Mainframe Connect Client Option.

client request An RPC or language request sent by a client to a server.

client/server An architecture in which the client is an application that handles the user interface and local data manipulation functions, and the server is an application providing data processing access and management. See also **client application**.

Client Services Application A customer-written CICS program initiated on the host that uses the API to invoke the Mainframe Connect Client Option as a client to the DirectConnect server or to ASE. See also **application program interface**, **Client Services for CICS**.

Client Services for CICS
A Sybase host API that invokes the Mainframe Connect Server Option as a client to an access service for DB2 UDB or ASE. See also **application program interface**, **Customer Information Control System**, **Client Services Application**, **Mainframe Connect Server Option**.

clustered index
An index in which the physical order and the logical (indexed) order is the same. Compare with **nonclustered index**.

code page
An assignment of graphic characters and control function meanings to all code points.

commit
A process that makes permanent all changes made to one or more database files since the initiation of the application program, the start of an interactive session, or the last commit or rollback operation. Compare with **rollback**.

Common Programming Interface
Specifies the languages and services used to develop applications across SAA environments. The elements of the CPI specification are divided into two parts: processing logic and services.

configuration file
A file that specifies the characteristics of a system or subsystem.

configuration set
A section into which service library configuration files are divided.

conversion
The transformation between values that represent the same data item but which belong to different datatypes. Information can be lost due to conversion, because accuracy of data representation varies among different datatypes.

connection
A network path between two systems. For SNA, the path connects a logical unit (LU) on one machine to an LU on a separate machine. For TCP/IP, the path connects TCP modules on separate machines.

connection router
A program provided with Mainframe Connect Client Option that directs requests to particular remote servers. Mainframe system programmers use the connection router to define remote servers and server connections to Mainframe Connect Client Option.

Connection Router Table
A memory-resident table maintained by a Mainframe Connect Client Option system programmer that lists servers and the connections that a Client-Library transaction can use to access them.

control section
The part of a program specified by the programmer to be a relocatable unit, all elements of which are to be loaded into adjoining main storage locations.

control statement
In programming languages, a statement that is used to alter the continuous sequential execution of statements. A control statement can be a conditional statement or an imperative statement.

conversation-level security — The passing of client login information to the mainframe by TRS when it allocates a conversation.

CSA — See **Client Services Application**.

CSP — See **catalog stored procedure**.

cursor — In SQL, a named control structure used by an application program to point to a row of data.

Customer Information Control System — An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs.

DASD — See **direct access storage device**.

data definition statement — An IBM mainframe statement used to relate a name with a file.

data definition language — A language for describing data and data relationships in a database.

data set name — The term or phrase used to identify a data set.

database management system — The term or phrase to identify a data set.A computer-based system for defining, creating, manipulating, controlling, managing, and using databases.

database operation — A single action against the database. For Mainframe Connect DirectConnect for z/OS Option, a database operation is usually a single SQL statement. One or more database actions can be grouped together to form a request. See also **request**.

Database 2 — An IBM relational database management system.

datatype — A keyword that identifies the characteristics of stored information on a computer.

DB-Library — A Sybase and Microsoft API that allows client applications to interact with ODS applications. See also **application program interface**.

DBMS — See **database management system**.

DB2 UDB — See **Database 2**.

DDL — See **data definition language**.

DD statement — See **data definition statement**.

default language — The language that displays a user's prompts and messages.

destination-template transfer A transfer method in which source data is briefly put into a template where the user can specify that some action be performed on it before execution against a target database. See also **transfer**. Compare with **bulk copy transfer** and **express transfer**.

direct access storage device A device in which access time is effectively independent of the location of the data.

direct request A request sent directly from a client workstation through Transaction Router Service to the DirectConnect server without going through ASE. Contract with **indirect request**.

direct resolution A type of service name resolution that relies upon a client application specifying the exact name of the service to be used. See also **service name resolution**. Compare with **service name redirection**.

DirectConnect Manager A Java application from Sybase that can be used in Windows and UNIX environments. It provides remote management capabilities for DirectConnect products, including starting, stopping, creating, and copying services.

DirectConnect server The component of Mainframe Connect DirectConnect for z/OS Option that provides general management and support functions to service libraries.

dll See **dynamic link library**.

DSN See **data set name**.

dynamic link library A file containing executable code and data bound to a program at load time or runtime, rather than during linking.

dynamic SQL 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. Contrast with **static SQL**.

ECDA See **Enterprise Connect Data Access**.

ECDA Option for ODBC A Sybase solution that allows client applications to access ODBC data. It combines the functionality of the ECDA Option for ODBC architecture with ODBC to provide dynamic SQL access to target data, as well as the ability to support stored procedures and text and image pointers.

ECDA Option for Oracle A Sybase solution that provides Open Client access to Oracle databases. When used in combination with ASE, it provides many of the features of a distributed database system, such as location transparency, copy transparency, and distributed joins.

embedded SQL SQL statements that are embedded within a program and are prepared in the process before the program runs. After it is prepared, the statement itself does not change, although values of host variables specified within the statement might change.

end user A person who connects to a DirectConnect server using an application to access databases and perform transfers. See also **transfer**.

Enterprise Connect Data Access An integrated set of software applications and connectivity tools that allow access to data within a heterogeneous database environment, such as a variety of LAN-based, non-Sybase data sources, as well as mainframe data sources.

environment variable A variable that describes how an operating system runs and the devices it recognizes.

exit routine A user-written routine that receives control at predefined user exit points.

express transfer A form of bulk copy transfer that uses ODBC bulk APIs to improve performance when transferring bulk data between data sources. Because it uses the same syntax as bulk copy transfer, no modification of applications is required.

external call interface A CICS client facility that allows a program to call a CICS application as if the calling program had been linked synchronously from a previous program instead of started from a terminal.

External Security Manager An add-on security package for the z/OS mainframe, licensed by Computer Associates.

FCT See **forms control table**.

forms control table An object that contains the special processing requirements for output data streams received from a host system by a remote session.

gateway Connectivity software that allows two or more computer systems with different network architectures to communicate.

Gateway-Library A library of communication, conversion, tracing, and accounting functions supplied with Mainframe Connect Server Option.

globalization The combination of internationalization and localization. See **internationalization**, **localization**.

global variable A variable defined in one portion of a computer program and used in at least one other portion of the computer program. Contrast with **local variable**.

handler A routine that controls a program's reaction to specific external events, for example, an interrupt handler.

host The mainframe or other machine on which a database, an application, or a program resides. In TCP/IP, this is any system that is associated with at least one Internet address. See also **Transmission Control Protocol/Internet Protocol**.

host ID In Mainframe Connect Server Option, the ID that the TRS passes to the mainframe with a client request. The host ID is part of the client login definition at the TRS.

host password In Mainframe Connect Server Option, the password that the client passes to the mainframe with a client request.

host request library A DB2 UDB table that contains host-resident SQL statements that can be executed dynamically. See also **host-resident request**.

host-resident request A SQL request that resides in a DB2 UDB table called the host request library. See also **host request library**.

IMS See **Information Management System**.

indirect request A client request that is routed through a stored procedure on a SQL Server, which forwards the request to TRS as an RPC. Compare with **direct request**.

Information Management System A database/data communication system that can manage complex databases and networks.

interfaces file An operating system file that determines how the host client software connects to a Sybase product. An *interfaces* file entry contains the name of any DirectConnect server and a list of services provided by that server.

internationalization The process of extracting locale-specific components from the source code and moving them into one or more separate modules, making the code culturally neutral so it can be localized for a specific culture. See also **globalization**. Compare with **localization**.

keyword A word or phrase reserved for exclusive use by Transact-SQL.

language RPC The name TRS uses to represent a client's language request. TRS treats a language request as a remote procedure call (RPC) and maps it to a language transaction at the remote server.

language transaction
The server transaction that processes client language requests. The Mainframe Connect DB2 UDB Option language transaction for CICS is AMD2, which uses the DB2 UDB dynamic SQL facilities to process incoming SQL strings. The Mainframe Connect DB2 UDB Option for IMS uses SYRT by default.

linkage
In computer security, combining data or information from one information system with data or information from another system with the intention to derive additional information; for example, the combination of computer files from two or more sources.

linkage editor
A computer program that creates load modules from one or more object modules or creates load modules by resolving cross references among the modules, and if necessary, adjusts those addresses.

link-edit
To create a loadable computer program by using a linkage editor. See also **linkage editor**.

localization
The process of preparing an extracted module for a target environment, in which messages are displayed and logged in the user's language. Numbers, money, dates, and time are represented using the user's cultural convention, and documents are displayed in the user's language. See also **globalization**.

local variable
A variable that is defined and used only in one specified portion of a computer program. Contrast with **global variable**.

logical unit
A type of network addressable unit that enables a network user to gain access to network facilities and communicate remotely. A connection between a TRS and a CICS region is a connection between logical units.

logical unit 6.2
A type of logical unit that supports general communication between programs in a distributed processing environment. See also **advanced program-to-program communication**.

login ID
In Mainframe Connect Server Option, the ID that a client user uses to log in to the system.

login packet
Client information made available to Mainframe Connect Server Option. The client program sets this information in a login packet and sends it to TRS, which forwards it to the mainframe.

long-running transaction
A transaction that accepts more than one client request. Whereas short transactions end the communication after returning results to a client, a long-running transaction can await and process another request. Compare with **short transaction**.

LU 6.2
See **logical unit 6.2**.

mainframe access products Sybase products that enable client applications to communicate with mainframes in a client/server environment. See **client/server**.

Mainframe Connect The Sybase product set that provides access to mainframe data.

Mainframe Connect Client Option A Sybase product that, using Client-Library, allows mainframe clients to send requests to SQL Server, Open Server, the Mainframe Connect DB2 UDB Option and Mainframe Connect Server Option. Mainframe Connect Client Option provides capability for the mainframe to act as a client to LAN-based resources in the CICS or the IMS and MVS environment.

Mainframe Connect DB2 UDB Option A Sybase mainframe solution that provides dynamic access to DB2 UDB data. It is available in the CICS or IMS environment. See also **Customer Information Control System**, **Database 2**, **Multiple Virtual Storage**.

Mainframe Connect DirectConnect for z/OS Option A Sybase Open Server application that provides access management for non-Sybase databases, copy management (transfer), and remote systems management.

Mainframe Connect Server Option A Sybase product that provides capability for programmatic access to mainframe data. It allows workstation-based clients to execute customer-written mainframe transactions remotely. It is available for the CICS and the IMS and MVS environments

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.

network protocol A set of rules governing the way computers communicate on a network.

nonclustered index An index that stores key values and pointers to data. Compare with **clustered index**.

null Having no explicitly assigned value. NULL is not equivalent to 0 or to blank.

ODBC See **Open Database Connectivity**.

ODS See **Open Data Services**.

Open Client A Sybase product that provides customer applications, third-party products, and other Sybase products with the interfaces required to communicate with Open Client and Open Server applications.

Open Data Services A product that provides a framework for creating server applications that respond to DB-Library clients.

Open Database Connectivity A Microsoft API that allows access to both relational and non-relational databases. See also **application program interface**.

Open Server — A Sybase product that provides the tools and interfaces required to create a custom server. Clients can route requests to the DirectConnect server through an Open Server configured to meet specific needs, such as the preprocessing of SQL statements.

parameter — A variable that is given a constant value for a specified application and can denote the application. Compare with **property**.

Partner Certification Reports — Sybase publications that certify third-party or Sybase products to work with other Sybase products.

Password Expiration Management — An IBM password management program with CICS Version 3.3 through an optional program temporary fix, and as an integral part of CICS with version 4.1 and higher.

PEM — See **Password Expiration Management**.

PL/1 — See **Programming Language /1**.

primary database — The database management system that the DirectConnect server is always connected to. It is implied in the transfer statement.

Programming Language/1 — A programming language designed for use in a wide range of commercial and scientific computer applications.

property — A setting for a server or service that defines the characteristics of the service, such as how events are logged. Compare with **parameter**.

protocol — The rules for requests and responses used to manage a network, transfer data, and synchronize the states of network components.

query — A request for data from a database, based upon specified conditions.

Registry — The part of the Windows operating system that holds configuration information for a particular machine.

relational database — A database in which data is viewed as being stored in tables consisting of columns (data items) and rows (units of information).

relational operators — Operators supported in search conditions.

relops — See **relational operators**.

remote procedure call — A call to execute a stored procedure on a remote server. For Mainframe Connect Server Option, an RPC is a direct request from a client to TRS. For Mainframe Connect Client Option, a Client-Library transaction that calls a procedure on a remote server acts like an RPC.

remote stored procedure — A customer-written CICS program using an API that resides on the mainframe and communicates with Mainframe Connect DB2 UDB Option. See also **Customer Information Control System**, **stored procedure**. Compare with **Client Services Application**.

remote systems management — A feature that allows a system administrator to manage multiple DirectConnect servers and multiple services from a client.

Replication Server — A Sybase SQL Server application that maintains replicated data and processes data transactions received from a data source.

request — One or more database operations an application sends as a unit to the database. Depending upon the response, the application commits or rolls back the request. See also **commit**, **rollback**, **unit of work**.

resource table — A main storage table that associates each resource identifier with an external logical unit (LU) or application program.

rollback — An instruction to a database to back out of changes requested in a unit of work. Compare with **commit**.

router — An attaching device that connects two LAN segments, which use similar or different architectures, at the Open System Interconnection (OSI) reference model network layer. Contrast with **gateway**.

RPC — See **remote procedure call.**

RSP — See **remote stored procedure**.

SAA — See **System Application Architecture**.

secondary connection — The connection specified in the transfer statement. It represents anything that can be accessed using Mainframe Connect Client Option, such as ASE or another access service.

secondary database — In transfer processing, the supported database that is specified in the transfer statement. Compare with **primary database**.

server — A functional unit that provides shared services to workstations over a network. See also **client/server**. Compare with **client**.

server process ID — A positive integer that uniquely identifies a client connection to the server.

service — A functionality available in Mainframe Connect DirectConnect for z/OS Option. It is the pairing of a service library and a set of specific configuration properties.

service library — In Mainframe Connect DirectConnect for z/OS Option, a set of configuration properties that determine service functionality. See also **access service library**, **administrative service library**, **Transaction Router Service library**, **transfer service library**.

service name redirection — A type of service name resolution that allows a system administrator to create an alternative mechanism to map connections with services. See also **service name resolution**. Compare with **direct resolution**.

service name redirection file — The default name of the file used for the service name redirection feature. See **service name redirection**.

service name resolution — The DirectConnect server mapping of an incoming service name to an actual service. See also **direct resolution**, **service name redirection**.

session — A connection between two programs or processes. In APPC communications, sessions allow transaction programs to have conversations between the partner LUs. See also **advanced program-to-program communication**.

short transaction — A mainframe transaction that ends the communication when it finishes returning results to the client. Compare with **long-running transaction**.

SNA — See **Systems Network Architecture**.

SNRF — See **service name redirection file**.

SPID — See **server process ID**.

SQL — See **structured query language**.

SQLDA — See **SQL descriptor area**.

sqledit — A utility for creating and editing *sql.ini* files and file entries.

sql.ini — The interfaces file containing definitions for each DirectConnect server to which a workstation can connect. The file must reside on every client machine that connects to ASE.

SQL descriptor area — A set of variables used in the processing of SQL statements.

SQL stored procedure — A single SQL statement that is statically bound to the database. See also **stored procedure**.

static SQL — SQL statements that are embedded within a program and prepared during the program preparation process before the program runs. Compare with **dynamic SQL**.

stored procedure A collection of SQL statements and optional control-of-flow statements stored under a particular name. Adaptive Server stored procedures are called "system procedures." See also **remote stored procedure**, **system procedures**.

structured query language An IBM industry-standard language for processing data in a relational database.

stub A program module that transfers remote procedure calls (RPCs) and responses between a client and a server.

SYRT The component of Mainframe Connect DB2 UDB for IMS that allows clients to submit SQL language requests to DB2 through IMS.

System Administrator The person in charge of server system administration, including installing and maintaining DirectConnect servers and service libraries.

System Application Architecture An IBM proprietary plan for the logical structure, formats, protocols, and operational sequences for transmitting information units through networks and controlling network configuration and operation. See also **advanced program-to-program communication**.

system procedures A stored procedure that ASE supplies for use in system administration. System procedures serve as shortcuts for retrieving information from system tables, or a mechanism for accomplishing database administration. See also **stored procedure**.

Systems Network Architecture An IBM proprietary plan for the structure, formats, protocols, and operational sequences for transmitting information units through networks. See also **advanced program-to-program communication**.

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 A Sybase application-level protocol that defines the form and content of relational database requests and replies.

target A system, program, or device that interprets, rejects, satisfies, or replies to requests received from a source.

target database The database to which the DirectConnect server transfers data or performs operations on specific data.

TCP/IP See **Transmission Control Protocol/Internet Protocol**.

TDS See **Tabular Data Stream**.

transaction A unit of processing initiated by a single request. A transaction consists of one or more application programs that, when executed, accomplish a particular action. In Mainframe Connect Server Option, a client request (RPC or language request) invokes a mainframe transaction. In Mainframe Connect Client Option, a mainframe transaction executes a stored procedure on a remote server.

transaction processing A sequence of operations on a database that is viewed by the user as a single, individual operation.

Transaction Router Service A Mainframe Connect DirectConnect for z/OS Option program used when the mainframe acts as a transaction server to route requests from remote clients to the Mainframe Connect Server Option and return results to the clients.

Transaction Router Service library A service library that facilitates access to remote transactions, allowing customers to execute transactions from virtually any mainframe data source. See also **service library**.

Transact-SQL A Sybase-enhanced version of the SQL database language used to communicate with ASE.

transfer A Mainframe Connect DirectConnect for z/OS Option feature that allows users to move data or copies of data from one database to another.

transfer service library A service library that provides copy management functionality. See also **service library**.

Transmission Control Protocol/Internet Protocol A set of communication protocols that supports peer-to-peer connectivity functions for both local and wide area networks.

trigger A form of stored procedure that automatically executes when a user issues a change statement to a specified table.

TRS See **Transaction Router Service**.

TRS library See **Transaction Router Service library**.

T-SQL See **Transact-SQL**.

unit of work One or more database operations grouped under a commit or rollback. A unit of work ends when the application commits or rolls back a series of requests, or when the application terminates. See also **commit**, **rollback**, **transaction**.

user ID User identification. The ID number by which a user is known in a specific database or system.

variable An entity that is assigned a value. Mainframe Connect DirectConnect for z/OS Option has two kinds of variables: *local* and *global.*

view An alternate representation of data from one or more tables. A view can include all or some of the columns contained the table or tables on which it is defined.

Virtual Storage Access Method An IBM-licensed program that controls communication and the flow of data in an SNA network.

Virtual Telecommunications Access Method IBM mainframe software that allows communication on an SNA network between mainframes and allows the mainframe to have multiple sessions per connection.

VSAM See **Virtual Storage Access Method.**

VTAM See **Virtual Telecommunications Access Method**.

wildcard A special character that represents a range of characters in a search pattern.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W