SYBASE®

プログラマーズ・ガイド補足

Open Client™/Open Server™

15.5

[Microsoft Windows 版]

ドキュメント ID：DC35454-01-1550-01
改訂：2009 年 11 月

マニュアルの注文

マニュアルの注文を承ります。ご希望の方は、サイベース株式会社営業部または代理店までご連絡ください。マニュアルの変更は、弊社の定期的なソフトウェア・リリース時にのみ提供されます。

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

目次

はじめに

このマニュアルは、Open Client™ および Open Server™ の『リファレンス・マニュアル』および『プログラマーズ・ガイド』の補足です。Open Client/Server 製品を使用するアプリケーションの作成、設定、およびトラブルシューティングのためのプラットフォーム固有の情報を提供します。

このマニュアルでは、特に明記しないかぎり、すべての Microsoft Windows プラットフォームを “Windows” と表記します。

対象読者

このマニュアルは、次の方を対象としています。

- Open Client/Server 製品を使用して、Sybase® およびサード・パーティのアプリケーションを作成するデスクトップ・アプリケーション開発者
- bcp、defncopy、および isql ユーティリティについての情報を必要とする方
- cpre および cobpre プリコンパイラについての情報を必要としている方

このマニュアルの内容

このマニュアルには、以下の章があります。

- 「第 1 章 Open Client と Open Server のアプリケーションの構築」では、Open Client ライブラリと Open Server ライブラリを使用するアプリケーションを構築するための情報を提供します。
- 「第 2 章 Client-Library/C のサンプル・プログラム」では、Client-Library サンプル・プログラムのロケーションや内容などについての情報を提供します。
- 「第 3 章 Open Client DB-Library/C のサンプル・プログラム」では、DB-Library™ サンプル・プログラムのロケーションや内容などについての情報を提供します。
- 「第 4 章 Open Server Server-Library/C のサンプル・プログラム」では、Server-Library サンプル・プログラムのロケーションや内容などについての情報を提供します。
- 「第 5 章 Open Client Embedded SQL/C」では、Embedded SQL/C サンプル・プログラム、実行プログラムの構築、およびアプリケーションのコンパイルとリンクについての情報を提供します。

- 「第 6 章 Open Client Embedded SQL/COBOL」では、Embedded SQL/Cobol サンプル・プログラム、実行プログラムの構築、およびアプリケーションのコンパイルとリンクについての情報を提供します。
- 「付録 A ユーティリティ・コマンド・リファレンス」には、Open Client に関連するコマンドとユーティリティの構文、パラメータ、識別子の詳細を説明するリファレンス・ページが含まれます。

関連マニュアル

詳細については、次のマニュアルを参照できます。

- 『Open Client Client-Library/C リファレンス・マニュアル』では、Open Client Client-Library のリファレンス情報について説明しています。
- 『Open Client および Open Server Common Libraries リファレンス・マニュアル』には、CS-Library のリファレンス情報が記載されています。CS-Library は、Client-Library と Server-Library の両方のアプリケーションで役立つユーティリティ・ルーチンの集まりです。
- 『Open Client Client-Library/C プログラマーズ・ガイド』では、Client-Library アプリケーションの設計方法および実装方法について説明しています。
- 『Open Client DB-Library/C リファレンス・マニュアル』では、Open Client DB-Library のリファレンス情報について説明しています。
- 『Open Server Server-Library/C リファレンス・マニュアル』には、Open Server Server-Library のリファレンス情報が記載されています。
- 『Open Client Embedded SQL/C プログラマーズ・ガイド』では、Embedded SQL/C アプリケーションの設計方法および実装方法について説明しています。
- 『Open Client Embedded SQL/COBOL プログラマーズ・ガイド』では、Embedded SQL/COBOL アプリケーションの設計方法および実装方法について説明しています。

次の情報については、『Open Client/Server 設定ガイド Windows 版』を参照してください。

- Open Client アプリケーションとサーバ間で通信するために環境を設定する方法
- Sybase アプリケーションをローカライズする方法

Open Server および Software Developer's Kit で利用可能な新機能については、Windows、Linux、UNIX、および Mac OS X 版の『Open Server および SDK 新機能』を参照してください。このマニュアルは、新機能の提供に伴って改訂されます。

その他の情報

Sybase Getting Started CD、SyBooks™ CD、Sybase Product Manuals Web サイトを利用すると、製品について詳しく知ることができます。

- Getting Started CD には、PDF 形式のリリース・ノートとインストール・ガイド、SyBooks CD に含まれていないその他のマニュアルや更新情報が収録されています。この CD は製品のソフトウェアに同梱されています。Getting Started CD に収録されているマニュアルを参照または印刷するには、Adobe Acrobat Reader が必要です (CD 内のリンクを使用して Adobe の Web サイトから無料でダウンロードできます)。
- SyBooks CD には製品マニュアルが収録されています。この CD は製品のソフトウェアに同梱されています。Eclipse ベースの SyBooks ブラウザを使用すれば、使いやすい HTML 形式のマニュアルにアクセスできます。

  一部のマニュアルは PDF 形式で提供されています。これらのマニュアルは SyBooks CD の PDF ディレクトリに収録されています。PDF ファイルを開いたり印刷したりするには、Adobe Acrobat Reader が必要です。

  SyBooks をインストールして起動するまでの手順については、Getting Started CD の『SyBooks インストール・ガイド』、または SyBooks CD の *README.txt* ファイルを参照してください。
- Sybase Product Manuals Web サイトは、SyBooks CD のオンライン版であり、標準の Web ブラウザを使ってアクセスできます。また、製品マニュアルのほか、EBFs/Maintenance、Technical Documents、Case Management、Solved Cases、ニュース・グループ、Sybase Developer Network へのリンクもあります。

  Sybase Product Manuals Web サイトは、Product Manuals にあります。(http://www.sybase.com/support/manuals/)

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品認定の最新情報にアクセスする**

1. Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)
2. [Partner Certification Report] をクリックします。
3. [Partner Certification Report] フィルタで製品、プラットフォーム、時間枠を指定して [Go] をクリックします。
4. [Partner Certification Report] のタイトルをクリックして、レポートを表示します。

❖ **コンポーネント認定の最新情報にアクセスする**

1 Web ブラウザで Availability and Certification Reports を指定します。(http://certification.sybase.com/)

2 [Search By Base Product] で製品ファミリとベース製品を選択するか、[Search by Platform] でプラットフォームとベース製品を選択します。

3 [Search] をクリックして、入手状況と認定レポートを表示します。

❖ **Sybase Web サイト (サポート・ページを含む) の自分専用のビューを作成する**

MySybase プロファイルを設定します。MySybase は無料サービスです。このサービスを使用すると、Sybase Web ページの表示方法を自分専用にカスタマイズできます。

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 [MySybase] をクリックし、MySybase プロファイルを作成します。

Sybase EBF とソフトウェア・メンテナンス

❖ **EBF とソフトウェア・メンテナンスの最新情報にアクセスする**

1 Web ブラウザで Sybase Support ページを指定します。(http://www.sybase.com/support)

2 [EBFs/Maintenance] を選択します。MySybase のユーザ名とパスワードを入力します。

3 製品を選択します。

4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースの一覧が表示されます。

鍵のアイコンは、「Technical Support Contact」として登録されていないため、一部の EBF/Maintenance リリースをダウンロードする権限がないことを示しています。未登録でも、Sybase 担当者またはサポート・コンタクトから有効な情報を得ている場合は、[Edit Roles] をクリックして、「Technical Support Contact」の役割を MySybase プロファイルに追加します。

5 EBF/Maintenance レポートを表示するには [Info] アイコンをクリックします。ソフトウェアをダウンロードするには製品の説明をクリックします。

表記規則

表 1 では構文の表記規則について説明します。

表 *1:* 構文の表記規則

キー	定義
command	コマンド名、コマンドのオプション名、ユーティリティ名、ユーティリティのフラグ、キーワードは sans serif で示す。
variable	変数 (ユーザが入力する値を表す語) は斜体で表記する。
{ }	中カッコは、その中から必ず 1 つ以上のオプションを選択しなければならないことを意味する。コマンドには中カッコは入力しない。
[ ]	角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。
( )	カッコはコマンドの一部として入力する。
|	中カッコまたは角カッコの中の縦線で区切られたオプションのうち 1 つだけを選択できることを意味する。
,	中カッコまたは角カッコの中のカンマで区切られたオプションをいくつでも選択できることを意味する。複数のオプションを選択する場合には、オプションをカンマで区切る。

アクセシビリティ機能

このマニュアルには、アクセシビリティを重視した HTML 版もあります。この HTML 版マニュアルは、スクリーン・リーダーで読み上げる、または画面を拡大表示するなどの方法により、その内容を理解できるよう配慮されています。

Open Client および Open Server のマニュアルは、連邦リハビリテーション法第 508 条のアクセシビリティ規定に準拠していることがテストにより確認されています。第 508 条に準拠しているマニュアルは通常、World Wide Web Consortium (W3C) の Web サイト用ガイドラインなど、米国以外のアクセシビリティ・ガイドラインにも準拠しています。

注意 アクセシビリティ・ツールを効率的に使用するには、設定が必要な場合もあります。一部のスクリーン・リーダーは、テキストの大文字と小文字を区別して発音します。たとえば、すべて大文字のテキスト (ALL UPPERCASE TEXT など) はイニシャルで発音し、大文字と小文字の混在したテキスト (Mixed Case Text など) は単語として発音します。構文規則を発音するようにツールを設定すると便利かもしれません。詳細については、ツールのマニュアルを参照してください。

Sybase のアクセシビリティに対する取り組みについては、Sybase Accessibility (http://www.sybase.com/accessibility) を参照してください。Sybase Accessibility サイトには、第 508 条と W3C 標準に関する情報へのリンクもあります。

不明な点があるときは

Sybase ソフトウェアがインストールされているサイトには、Sybase 製品の保守契約を結んでいるサポート・センタとの連絡担当の方 (コンタクト・パーソン) を決めてあります。マニュアルだけでは解決できない問題があった場合には、担当の方を通して Sybase のサポート・センタまでご連絡ください。

第 1 章

Open Client と Open Server のアプリケーションの構築

この章では、Windows プラットフォーム上で Open Client ライブラリと Open Server ライブラリを使用するアプリケーションを構築するために必要な情報について説明します。また、Sybase ライブラリを使用して Windows の実行プログラムを構築するための条件について説明します。

Open Client と Open Server の使用条件

Windows プラットフォームで Open Client と Open Server のアプリケーションをコンパイルしてリンクするには、次のような条件が必要です。

- ANSI 準拠の C コンパイラがインストールされている。
- INCLUDE および LIB 環境変数を定義する。
- PATH 環境変数が *%SYBASE%¥%SYBASE_OCS%¥dll* ディレクトリをインクルードするよう設定する。

サポートされるプラットフォームの詳細については、『Open Server および SDK 新機能』(各 Windows、Linux、UNIX、Mac OS X 版) を参照してください。

C コンパイラ

サンプル・プログラムを使用したり、アプリケーションを構築したりするには、ANSI 準拠の C コンパイラがインストールされている必要があります。Sybase では、Microsoft Visual C++ バージョン 6.0 について動作を確認しています。他のコンパイラについても Sybase によって動作確認されている場合があります。動作確認されているコンパイラの最新のリストについては、Sybase の営業担当者に問い合わせてください。

Open Client および Open Server プログラムのコンパイルと実行の方法は、他の C 言語プログラムと同じです。アプリケーションをコンパイルしてリンクする方法については、使用するコンパイラのマニュアルを参照してください。

警告！ Sybase が動作確認をしていない ANSI 準拠の C コンパイラを使用して問題が発生した場合は、Sybase が動作確認しているコンパイラを使用して問題を再現できる場合にのみ、Sybase からテクニカル・サポートを受けることができます。

サポートされる Windows プラットフォームの詳細については、『Open Server および SDK 新機能』(各 Windows、Linux、UNIX、Mac OS X 版) を参照してください。

環境変数とヘッダ・ファイル

アプリケーションを正しく機能させるには、いくつかの環境変数を設定します。

表 *1-1:* 環境変数の説明

変数	説明
INCLUDE	*%SYBASE%¥%SYBASE_OCS%¥include* ディレクトリを指すディレクトリ・パスを設定します。インストールが完了すると、このディレクトリにはヘッダ・ファイルが格納されます。
LIB	*%SYBASE%¥%SYBASE_OCS%¥lib* ディレクトリを指すディレクトリ・パスを設定します。インストールが完了すると、このディレクトリにはインポート・ライブラリ・ファイルが格納されます。
PATH	*%SYBASE%¥%SYBASE_OCS%¥dll* ディレクトリを指すディレクトリ・パスを設定します。インストールが完了すると、このディレクトリには Sybase DLL が格納されます。

ヘッダ・ファイル

使用している製品によっては、Open Client および Open Server アプリケーションをコンパイルするときに、1 つ以上のヘッダ・ファイルをインクルードする必要がある場合があります。

DB-Library ヘッダ・ファイル

- *sybdb.h* – DB-Library ルーチンで使用される定義と型定義を含んでいる。*sybdb.h* は、『Open Client DB-Library/C リファレンス・マニュアル』の説明どおりに使用する。*sybdb.h* ファイルには、他に必要なすべてのヘッダ・ファイルがインクルードされている。
- *sybfront.h* –『Open Client DB-Library/C リファレンス・マニュアル』で説明されている関数の戻り値と、STDEXIT および ERREXIT などの記号定数を定義している。また、*sybfront.h* には、プログラム変数の宣言に使用できるデータ型のための型定義も含まれている。
- *syberror.h* – エラー重大度の値を含んでいる。プログラムがこれらの値を参照する場合はインクルードする必要がある。

Client-Library ヘッダ・ファイル

- *ctpublic.h* ― すべての Client-Library アプリケーションで必須。このファイルには、他に必要なすべてのヘッダ・ファイルがインクルードされている。
- *bkpublic.h* ― Bulk-Library を呼び出すアプリケーションの場合は必須。このファイルには、他に必要なすべてのヘッダ・ファイルがインクルードされている。

Server-Library ヘッダ・ファイル

- *ospublic.h* ― すべての Server-Library アプリケーションで必須。このファイルには、他に必要なすべてのヘッダ・ファイルがインクルードされている。
- *bkpublic.h* ― Bulk-Library を呼び出すアプリケーションの場合は必須。このファイルには、他に必要なすべてのヘッダ・ファイルがインクルードされている。

インポート・ライブラリと DLL (ダイナミック・リンク・ライブラリ)

この項では、インポート・ライブラリと DLL (ダイナミック・リンク・ライブラリ) について説明します。

インポート・ライブラリ

Open Client と Open Server のインポート・ライブラリには、Open Client や Open Server のアプリケーションを構築するためにリンカで使用される情報が含まれています。表 1-2 は、アプリケーションをコンパイルしてリンクするときにインクルードするインポート・ライブラリを示します。

表 *1-2: Open Client と Open Server のインポート・ライブラリ*

DB-Library インポート・ライブラリ	Client-Library インポート・ライブラリ	Server-Library インポート・ライブラリ
libsybdb.lib ― DB-Library	*libsybct.lib* ― Client-Library *libsybcs.lib* – CS-Library *libsybblk.lib* – Bulk-Library Bulk-Library 呼び出しを使用する場合にだけ、Bulk-Library インポート・ライブラリ *libsybblk.lib* とリンクする。	*libsybsrv.lib* – Server-Library *libsybct.lib* – Client-Library *libsybcs.lib* – CS-Library *libsybblk.lib* – Bulk-Library Bulk-Library 呼び出しを使用する場合にだけ、Bulk-Library インポート・ライブラリ *libsybblk.lib* とリンクする。

DLL (ダイナミック・リンク・ライブラリ)

Windows の Open Client と Open Server ライブラリのアプリケーションは、実行時に Open Client DLL 内の関数を呼び出す必要があります。Sybase DLL がパスに含まれていることを確認してください。PATH 環境変数に *%SYBASE%¥%SYBASE_OCS%¥dll* ディレクトリを指定してください。表 1-3 は、Open Client と Open Server のライブラリに含まれる DLL を示します。

表 *1-3: Open Client と Open Server の DLL*

DB-Library の DLL	Client-Library の DLL	Server-Library の DLL
libsybdb.dll – DB-Library *libsybintl.dll* – ローカライゼーション・サポート・ライブラリ *libsybtcl.dll* – トランスポート制御レイヤ *libsybcomn.dll* – 内部共通ライブラリ *libsybunic.dll* – Unicode-Library	*libsybct.dll* – Client-Library *libsybcs.dll* – CS-Library *libsybintl.dll* – ローカライゼーション・サポート・ライブラリ *libsybtcl.dll* – トランスポート制御レイヤ *libsybcomn.dll* – 内部共通ライブラリ *libsybblk.dll* – Bulk-Library *libsybunic.dll* – Unicode-Library	*libsybct.dll* – Client-Library *libsybcs.dll* – CS-Library *libsybintl.dll* – ローカライゼーション・サポート・ライブラリ *libsybtcl.dll* – トランスポート制御レイヤ *libsybcomn.dll* – 内部共通ライブラリ *libsybsrv.dll* – Server-Library *libsybblk.dll* – Bulk-Library *libsybunic.dll* – Unicode-Library

設定条件

サンプル・プログラムおよび使用するアプリケーションが正しく動作するためには、次の条件が満たされている必要があります。

- SYBASE 環境変数が定義されている。
- *sql.ini* ファイルに、Open Client アプリケーションで使用されるサーバ名に対するクエリ・エントリが存在する。
- *sql.ini* ファイルに、Open Server アプリケーションで使用されるサーバ名に対するマスタ・エントリが存在する。
- Windows プラットフォームに最小限 64MB のメモリがある。

注意 SYBASE 環境変数の設定と *sql.ini* ファイルの設定の詳細については、『Open Client/Server 設定ガイド Windows 版』を参照してください。

プラットフォーム固有のデフォルト値

表 1-4 は、プラットフォーム固有のデフォルト値を持つ Open Client と Open Server のプロパティを示します。

表 *1-4: Client-Library* のプラットフォーム固有のプロパティ

ライブラリ	プロパティ名	説明	Windows でのデフォルト値
Client-Library Server-Library	CS_IFILE	*sql.ini* ファイルのパスと名前	SYBASE 環境変数で定義される *%SYBASE%¥ini* ディレクトリ内の *sql.ini* ファイル
	CS_MAX_CONNECT	このコンテキストでの最大接続数	25
	CS_PACKETSIZE	TDS パケット・サイズ	512 バイト
DB-Library	DBSETFILE	*sql.ini* ファイルのパスと名前	SYBASE 環境変数で定義される *%SYBASE%¥ini* ディレクトリ内の *sql.ini* ファイル
	DBSETMAXPROS	このコンテキストでの最大接続数	25

Client-Library のプログラミングについて

この項では、Windows プラットフォームでの特定の Client-Library ルーチンの動作と、『Open Client Client-Library/C リファレンス・マニュアル』および『Open Client Client-Library/C プログラマーズ・ガイド』でのそれらの説明との違いについて説明します。

ct_callback

ct_callback を使用して Client-Library に登録される Client-Library アプリケーション・ルーチンは、すべて CS_PUBLIC として宣言しなければなりません。宣言の例については、サンプル・ディレクトリにある *exutils.c* 内の ex_clientmsg_cb を参照してください。

デバッグ DLL の使い方

インストール時に選択したオプションによっては、Client-Library のデバッグ・バージョンと非デバッグ・バージョンの両方の *libsybct.dll* をインストールできます。デバッグ・バージョンの DLL は、Sybase *dll* ディレクトリの *debug* サブディレクトリに、非デバッグ・バージョンは *nondebug* サブディレクトリにあります。使用する方のバージョンを Sybase インストール・ディレクトリの *dll* サブディレクトリにコピーしてください。アプリケーションは Sybase インストール・ディレクトリの *dll* サブディレクトリ内の DLL を自動的に使用します。ct_debug は、デバッグ・バージョンの *libsybct.dll* を使用する場合にだけ機能します。

Client-Library のデバッグ・バージョンの詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

マルチスレッドのサポート

Client-Library バージョン 11.1 以降では、マルチスレッド・アプリケーションの開発に使用される Windows プラットフォームのスレッド・ライブラリがサポートされます。マルチスレッド・アプリケーションを作成する方法については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

コンパイルとリンクの実行例

Client-Library サンプル・プログラムを構築するための *makefile* は、*%SYBASE%¥%SYBASE_OCS%¥sample¥ctlib* ディレクトリに格納されています。この項では、Microsoft Visual C/C++ コンパイラ (バージョン 6.0) で使用できる、Windows 版 Client-Library アプリケーション用の *makefile* の例を示します。

```
#################################################################
# Microsoft makefile for sample programs
#
#################################################################
MAKEFILE=MAKEFILE

!ifndef SYBASE
SYBASEHOME=c:¥sybase
!else
SYBASEHOME=$(SYBASE)
!endif

COMPILE_DEBUG = 1

# Compiler AND linker flags
!ifdef COMPILE_DEBUG
CFLAGS = /W3 /MD /nologo /Zi /DWIN32
LFLAGS= /MAP /SUBSYSTEM:CONSOLE /DEBUG /DEBUGTYPE:cv
!else
CFLAGS = /W3 /MD /nologo /Od /DWIN32
LFLAGS= /MAP /SUBSYSTEM:CONSOLE
!endif
ASYNCDEFS = -DUSE_SIG_HANDLER=0
HDRS = example.h exutils.h
MTHDRS = example.h thrdutil.h thrdfunc.h
# Where to get includes and libraries
#
# SYBASE is the environment variable for sybase home directory
#
SYBINCPATH =   $(SYBASEHOME)¥$(SYBASE_OCS)¥include
BLKLIB =       $(SYBASEHOME)¥$(SYBASE_OCS)¥lib¥libsybblk.lib
CTLIB =        $(SYBASEHOME)¥$(SYBASE_OCS)¥lib¥libsybct.lib
CSLIB =       $(SYBASEHOME)¥$(SYBASE_OCS)¥lib¥libsybcs.lib
SYSLIBS= kernel32.lib advapi32.lib msvcrt.lib

# The generalized how to make an .obj rule
.c.obj:
        cl /I. /I$(SYBINCPATH) $(ASYNCDEFS) $(CFLAGS) -Fo$@ -c $<

all: exasync compute csr_disp getsend rpc blktxt i18n multthrd usedir firstapp
exconfig secct wide_rpc wide_dynamic wide_curupd wide_compute

uni: uni_firstapp uni_csr_disp uni_compute uni_blktxt uni_rpc
```

```
exasync compute csr_disp getsend rpc blktxt i18n multthrd usedir firstapp
exconfig secct twophase: $*.exe
            @echo Sample '$*' was built

wide_rpc wide_dynamic wide_curupd wide_compute: $*.exe
            @echo Sample '$*' was built

uni_firstapp uni_csr_disp uni_compute uni_blktxt uni_rpc: $*.exe
            @echo Sample '$*' was built

sample.exe: sample.obj $(MAKEFILE)
        link $(LFLAGS) -out:$*.exe sample.obj $(SYSLIBS)

exasync.exe: ex_alib.obj ex_amain.obj exutils.obj $(MAKEFILE)
        link $(LFLAGS) -out:$*.exe ex_alib.obj ex_amain.obj exutils.obj
$(SYSLIBS) $(CTLIB)$(CSLIB)
... compile and link lines for each Client-Library sample program goes here ...

clean:
            -del *.obj
            -del *.map
            -del *.exe
            -del *.err
            -del *.ilk
            -del *.pdb
```

この例では、

- Sybase ライブラリは Microsoft Windows x86 32 ビット・アプリケーション用に作成されています。
- SUBSYSTEM:CONSOLE はコンソール・アプリケーションを示しています。

詳細については、コンパイルとリンクに関する Microsoft の適切なマニュアルを参照してください。

DB-Library のプログラミングについて

この項では、Windows プラットフォームでの特定の DB-Library ルーチンの動作と、『Open Client DB-Library/C リファレンス・マニュアル』でのそれらの説明との違いについて説明します。

コンパイルとリンクの例

DB-Library/C アプリケーションをコンパイルしてリンクするためのコマンドの一般的な形式は次のとおりです。

```
!ifdef COMPILE_DEBUG
 CFLAGS = /W3 /MD /nologo /Z7
 LFLAGS= /MAP /SUBSYSTEM:CONSOLE /DEBUG /DEBUGTYPE:cv
 !else
 CFLAGS = /W3 /MD /nologo /Od
 LFLAGS= /MAP /SUBSYSTEM:CONSOLE
 !endif
```

Server-Library のプログラミングについて

この項では、Windows での特定の Server-Library ルーチンの動作と、『Open Server Server-Library/C リファレンス・マニュアル』でのそれらの説明との違いについて説明します。

srv_callback

srv_callback を使用して Server-Library に登録される Server-Library アプリケーション・ルーチンは、すべて CS_PUBLIC として宣言しなければなりません。宣言の例については、サンプル・ディレクトリにある *utils.c* 内の cs_err_handler を参照してください。

スケジューリング・モード

Windows で実行される Server-Library アプリケーションは、コルーチン・スケジューリング・モードまたはプリエンプティブ・スケジューリング・モードのいずれかで実行できます。コルーチン・スケジューリング (デフォルト) は、プリエンプティブ・スケジューリングをサポートしないほかのプラットフォームと互換性があります。プリエンプティブ・スケジューリング・モードを選択するには、srv_config 関数を使用して SRV_PREEMPT オプションを true に設定します。

プリエンプティブ・モード・プログラミング

プリエンプティブ・スケジューリング・モードでは、すべてのスレッドを同時に実行できます。スレッドのスケジューリングは Windows によって処理されます。プリエンプティブ・スケジューリングでは、1 つのスレッドがサーバを占有することはありません。プリエンプティブ・モードで実行する場合、アプリケーションはデバッガのスレッド機能を使用してスレッドを操作できます。この場合、サーバを停止させずにブロック処理オペレーションを実行することもできます。プリエンプティブ・モードでは、スレッド間で大量のデータを共有することがないため、Windows 上で稼働するアプリケーションのパフォーマンスが向上します。

注意 コルーチン・スケジューリングしか使用できないプラットフォームへの移植性を保証するには、Windows 固有のセマフォ API を使用するのではなく、常に Server-Library の mutex 機能を使用してグローバル・データを保護してください。

Windows 固有のプリエンプティブ・プログラミングでは、srv_sleep 呼び出しと srv_wakeup 呼び出しを使用します。

srv_sleep

次のコード例は、プリエンプティブ・モードでの srv_sleep の使い方を示します。

```
/*
 ** Request the mutex to prevent the logging service
 ** from calling srv_wakeup before srv_sleep is called.
 */
 if (WaitForSingleObject(Mutex,INFINITE) != WAIT_OBJECT_0)
         return(CS_FAIL);
 /*
 ** Send the log_request to the logging service.
 */
 if (srv_putmsgq(log_service,log_request, SRV_M_NOWAIT) == CS_FAIL)
         return(CS_FAIL);

 /*
 ** Sleep until the log service has processed the log request.
 */
 srv_sleep(log_request, LOGWAIT, NULL, NULL, (CS_VOID*)Mutex, (CS_VOID*)0);
```

srv_wakeup

mutex を使用してプリエンプティブ・モードで srv_sleep を使用する場合は、対応する srv_wakeup ルーチンの前に同じ mutex に対する要求がなければなりません。これによって、スリープしているスレッドは、srv_wakeup の実行に対する準備ができます。次のコード例は、プリエンプティブ・モードで使用される場合、srv_wakeup の前に mutex に対する要求がどのように置かれるかを示しています。

```
/*
 ** Loop forever, logging language text. srv_getmsg will cause
 ** this thread to be suspended until a message is available on
 ** the log_request message queue.
 */
 while((get_status = srv_getmsgq(msgqid, &log_request,
         SRV_M_WAIT, &info)) == CS_SUCCEED)
 {
      /*
       ** Do the logging here.
      */

     /*
     ** Request the mutex to make sure the sender
     ** has called srv_sleep.
     */
     if (WaitForSingleObject(Mutex,INFINITE) != WAIT_OBJECT_0)
             return(CS_FAIL);

         /*
         ** Wake up the thread that is waiting for the language
         ** text to be logged.
         */
         srv_wakeup(log_request, SRV_M_WAKE_FIRST, (CS_VOID*)0, (CS_VOID*)0);

     /*
     ** Release the mutex.
     */
     if (!ReleaseMutex(Mutex))
             return(CS_FAIL);
 }
```

コンパイルとリンクの実行例

Server-Library サンプル・プログラムを構築するための *makefile* は、*%SYBASE%¥%SYBASE_OCS%¥sample¥srvlib* ディレクトリに格納されています。次の例は、Microsoft Windows x86 32 ビット・アプリケーションをコンパイルしてリンクするための *makefile* の一部を示しています。

```
#############################################################################
#
# Microsoft makefile for building Sybase Open Server Samples for Windows
#
#############################################################################
MAKEFILE=MAKEFILE
!ifndef SYBASE
SYBASEHOME=c:¥sybase
!else
SYBASEHOME=$(SYBASE)¥$(SYBASE_OCS)
!endif

COMPILE_DEBUG = 1
# Compiler AND linker flags
!ifdef COMPILE_DEBUG
CFLAGS = /W3 /MD /nologo /Z7 /DWIN32
LFLAGS= /MAP /SUBSYSTEM:CONSOLE /DEBUG /DEBUGTYPE:cv
!else
CFLAGS = /W3 /MD /nologo /Od /DWIN32
LFLAGS= /MAP /SUBSYSTEM:CONSOLE
!endif
SYSLIBS = kernel32.lib advapi32.lib msvcrt.lib
SYBASELIBS = $(SYBASEHOME)¥lib¥libsybcs.lib $(SYBASEHOME)¥lib¥libsybct.lib
$(SYBASEHOME)¥lib¥libsybsrv.lib
BLKLIB = $(SYBASEHOME)¥lib¥libsybblk.lib
DBLIB = $(SYBASEHOME)¥lib¥libsybdb.lib
CTOSOBJ = args.obj attn.obj bulk.obj ¥
          connect.obj ctos.obj cursor.obj ¥
          dynamic.obj error.obj events.obj ¥
          language.obj mempool.obj options.obj ¥
          params.obj ¥
          rgproc.obj results.obj rpc.obj ¥
          send.obj shutdown.obj
all: lang fullpass ctos regproc ctwait version intlchar osintro multthrd secsrv
lang fullpass ctos regproc ctwait version intlchar osintro multthrd secsrv:
$*.exe
     @echo Sample '$*' was built
# The generalized how to make an .obj rule
.c.obj:
     cl /I. /I$(SYBASEHOME)¥include $(CFLAGS) -Fo$@ -c $<
lang.exe: lang.obj utils.obj
     link $(LFLAGS) -out:$*.exe $*.obj utils.obj $(SYSLIBS) $(SYBASELIBS)
fullpass.exe: fullpass.obj utils.obj
     link $(LFLAGS) -out:$*.exe $*.obj utils.obj $(SYSLIBS) $(SYBASELIBS)
```

```
ctos.exe: $(CTOSOBJ)
... compile and link lines for each Client-Library sample program goes here ...
clean:

      -del *.obj
      -del *.map
      -del *.exe
      -del *.err
/*
```

この例では、

- Sybase ライブラリは Microsoft Windows x86 32 ビット・アプリケーション用に作成されています。
- SUBSYSTEM:CONSOLE はコンソール・アプリケーションを示しています。

詳細については、コンパイルとリンクに関する Microsoft の適切なマニュアルを参照してください。

第 2 章

Client-Library/C のサンプル・プログラム

Open Client Client-Library は、クライアント・アプリケーションの作成に使用するルーチンの集まりです。Client-Library には、サーバにコマンドを送信するルーチンとそれらのコマンドの結果を処理するルーチンが含まれています。アプリケーション・プロパティの設定、エラー条件の処理、サーバとのアプリケーションの対話に関するさまざまな情報の提供を行うルーチンもあります。

Open Client に含まれている CS-Library は、Open Client アプリケーションや Open Server アプリケーションを作成するために使用できるユーティリティ・ルーチンの集まりです。Client-Library ルーチンは CS-Library 内で割り付けられる構造体を使用するため、すべての Client-Library アプリケーションには、CS-Library に対する呼び出しが少なくとも 1 つ含まれます。

Client-Library のサンプル・プログラムの使用

サンプル・プログラムは、Client-Library/C に固有な機能の例を示します。これらのプログラムは Client-Library/C のトレーニング用としてではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

注意 これらのサンプル・プログラムは、実際の運用環境で使用するために作成されているものではありません。運用環境で使用するプログラムには、エラーや特殊ケースの処理のためのコードを追加する必要があります。

Open Client サンプル プログラムを使用する前に、次の操作を実行します。

- SYBASE 環境変数に Sybase リリース・ディレクトリのパスを設定していない場合は、設定します。
- SYBASE_OCS 環境変数を、Open Client/Server 製品のホーム・ディレクトリに設定します。たとえば、Open Client/Server バージョン 15.5 のホーム・ディレクトリは、*OCS-15_0* です。
- DSQUERY 環境変数を接続先のサーバの名前 (*server_name*) に設定します。
- 付属の *makefile* を使用して make を実行し、*example_name* というサンプル実行プログラムを作成します。

使用する環境と変数の設定の詳細については、『Open Client/Server 設定ガイド Windows 版』を参照してください。

サンプル・プログラムのロケーション

Client-Library サンプル・プログラムは、*%SYBASE%¥%SYBASE_OCS%¥sample¥ctlib* にあり、これには以下が含まれています。

- サンプル・プログラムのソース・コード
- サンプル・プログラムのためのデータ・ファイル
- サンプル・プログラムのためのヘッダ・ファイル *example.h*

注意 サンプル・プログラムが常駐するディレクトリの内容のバックアップ・コピーを作成してください。これによって、もとのファイルの整合性に影響を与えずにサンプル・プログラムを使用することができます。

ヘッダ・ファイル

表 2-1 では、すべての Client-Library アプリケーションに必要なヘッダ・ファイルを説明しています。

表 *2-1: Client-Library* アプリケーションに必要なヘッダ・ファイル

ファイル	説明
ctpublic.h	Client-Library に対する呼び出しを含んでいるすべてのアプリケーション・ソース・ファイルで必要。このファイルの内容は次のとおり。 • Client-Library ルーチンで使用される記号定数の定義 • Client-Library ルーチンのための宣言
cspublic.h	CS-Library ヘッダ・ファイル。ファイルの内容は次のとおり。 • クライアント／サーバ共通の記号定数の定義 • クライアント／サーバ共通の構造体のための型定義 • CS-Library ルーチンの宣言
bkpublic.h	バルク・コピー・ルーチンに対する呼び出しを行うすべてのアプリケーション・ソース・ファイルで必要。
cstypes.h	Client-Library のデータ型のための型定義を含む。
sqlca.h	SQLCA 定義構造体のための型定義を含む。

example.h ファイル

すべてのサンプル・プログラムは、サンプル・ヘッダ・ファイル *example.h* を参照します。*example.h* の内容は、次のとおりです。

```
/*
** example.h
**
** This is the header file that goes with the Sybase
** Client-Library example programs.
*/
/*
** Define symbolic names, constants, and macros
*/
#define EX_MAXSTRINGLEN         255
#define EX_BUFSIZE              1024
#define EX_CTLIB_VERSION        CS_CURRENT_VERSION
#define EX_BLK_VERSION          BLK_VERSION_155
#define EX_ERROR_OUT            stderr
#define EX_BADVAL               (CS_INT)-1
#define EX_MAX_ARR              64

/*
** exit status values
*/
#define EX_EXIT_SUCCEED         0
#define EX_EXIT_FAIL            1
/*

** Define global variables used in all sample programs
*/
#define EX_SERVER       NULL    /* use DSQUERY env var */
```

```
#define EX_USERNAME     "sa"
#define EX_PASSWORD     ""

/*
** Uncomment the following line to test the HA Failover feature.
*/
/* #define HAFAILOVER 1 */
#define EX_SCREEN_INIT()
```

EX_USERNAME と EX_PASSWORD に次の変更を加えます。

- EX_USERNAME は、*example.h* 内で “sa” と定義されています。サンプル・プログラムを使用する前に、*example.h* を編集して “sa” をサーバのログイン名に変更します。
- EX_PASSWORD は、*example.h* で null (“ ”) 文字列として定義されています。サンプル・プログラムを使用する前に、*example.h* を編集して、この値をサーバのパスワードに変更できます。次の処理を実行できます。
  - サンプル・プログラムを実行している間だけ、サーバ・パスワードを null (“ ”) に変更します。ただし、この場合、セキュリティを侵害される可能性があります。つまり、承認されていないユーザでもサーバにログインできます。これでは問題がある場合は、他の 2 つの方法のどちらかを選択してください。
  - null (“ ”) 文字列を、使用するサーバのパスワードに変更します。オペレーティング・システムの保護メカニズムを使用して、使用中は他のユーザがヘッダ・ファイルにアクセスできないようにします。サンプル・プログラムの使用を終了した後、変更した行を “server_password” に戻します。
  - サンプル・プログラム内で、サーバのパスワードを設定する ct_con_props コードを削除して、ユーザにサーバのパスワードを要求するようにコードを変更します。このコードはプラットフォームに固有なので、Sybase からは提供されません。

サンプル・プログラムの概要

Client-Library ルーチンの一般的な使い方の例を示すサンプル・プログラムが提供されています。サンプル・プログラムには、Adaptive Server® サンプル・データベースを使用するものもあります。サンプル・データベースをインストールする方法については、『ASE インストール・ガイド』を参照してください。

サンプル・プログラムは C ソース・ファイルです。サンプル・プログラムを使用したり、アプリケーションを構築したりするには、使用するプラットフォーム上に適切なコンパイラをインストールする必要があります。

サンプル・プログラムのためのユーティリティ・ルーチン

exutils.c ファイルには、他のすべてのサンプル・プログラムで使用されるユーティリティ・ルーチンが含まれています。この exutils.c は、アプリケーションが、より高いレベルのプログラムから Client-Library の実装の詳細部分を隠す方法を示しています。

wide_util.c ファイルには、wide_* サンプル・プログラムで使用される次の一般的なルーチンが含まれています。

- init_db – コンテキストを割り付けて、ライブラリを初期化します。さらにコールバック ルーチンをインストールします。このルーチンは、いくつかのサンプル・プログラムの開始時に呼び出されます。
- cleanup_db – サーバとの接続をクローズして、コンテキスト構造をクリーンアップします。この関数は、*wide_curupd.c* および *wide_dynamic.c* サンプル・プログラムの終了時に呼び出されます。
- connect_db – サーバに接続して、適切なユーザ名とパスワードを設定します。
- handle_returns – 戻される結果タイプを処理します。
- fetch_n_print – バインドされるデータをフェッチしてホスト変数に格納します。

これらのルーチンの詳細については、サンプル・ソース・ファイル内の先頭にあるコメントを参照してください。

firstapp.c サンプル・プログラム

firstapp.c は、サーバに接続し、select クエリを送信して、ローを表示する初歩的な例です。このサンプル・プログラムについては、『Open Client Client-Library/C プログラマーズ・ガイド』の「第 1 章 Client-Library を使用する前に」を参照してください。

uni_firstapp.c サンプル・プログラム

uni_firstapp.c は、unichar データ型と univarchar データ型を使用するために *firstapp.c* を変更したものです。サーバに接続する初歩的なサンプル・プログラムであり、select クエリを送信し、ローを出力します。*firstapp.c* プログラムについては、『Open Client Client-Library/C プログラマーズ・ガイド』を参照してください。

arraybind.c サンプル・プログラム

arraybind.c は、ct_command により起動された CS_LANG_CMD とともに配列バインドを使用する方法を示します。このサンプル・プログラムは、pubs2 データベース内のハードコード・テーブルのハードコード・クエリを使用します。このクエリは、select 文を使用する言語コマンドによって定義されます。次に、*arraybind.c* は標準の ct_results while ループを使用して結果を処理します。カラム値をプログラム配列にバインドした後、標準の ct_fetch ループでローをフェッチして表示します。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

非同期処理のサンプル・プログラム

このサンプル・プログラムには *ex_alib.c* と *ex_amain.c* の 2 つのファイルがあり、Client-Library の上位に非同期レイヤを作成する方法を示します。このプログラムは、Client-Library によって提供される仕組みを使用して連続的なポーリングと、Client-Library の完了コールバックの使用を可能にします。

非同期処理のサンプル・プログラムは次の 2 つのファイルで構成されます。

- *ex_alib.c* － サンプル・プログラムのライブラリ部分のソース・コードを含んでいます。これは、非同期呼び出しをサポートするライブラリ・インタフェースの一部であることを意味します。このモジュールは、1 つの非同期オペレーション内でサーバにクエリを送信してサーバから結果を取り出す手段を提供します。
- *ex_amain.c* － *ex_alib.c* によって提供されるサービスを使用するメイン・プログラムのソース・コードが含まれています。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントと *EX_AREAD.ME* ファイルを参照してください。

blktxt.c サンプル・プログラム

サンプル・プログラム *blktxt.c* は、バルク・コピー・ルーチンを使用して静的データをサーバ・テーブルにコピーします。このプログラムでは、プログラム変数にバインドされている 3 つのローのデータが、1 つのバッチとしてサーバに送信されます。このローは、テキスト・データを送信するために blk_textxfer を使用してもう一度送信されます。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

compute.c サンプル・プログラム

compute.c は、計算結果の処理の例を示します。

- クエリを、言語コマンドを使用してサーバに送信します。
- 標準の ct_results while ループを使用して結果を処理します。
- カラム値をプログラム変数にバインドします。
- 標準の ct_fetch while ループでローをフェッチして表示します。

クエリは次のとおりです。

```
select type, price from titles
where type like "%cook"
order by type, price
compute sum(price) by type
compute sum(price)
```

このクエリは、通常のローと計算ローの両方を返します。計算ローは 2 つの compute 句によって生成されます。

- 最初の compute 句は、type の値が変化するたびに計算ローを生成します。

  ```
  compute sum(price) by type
  ```

- 2 つ目の compute 句は、最後に返される 1 つの計算ローを生成します。

  ```
  compute sum(price)
  ```

このプログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意　このサンプル・プログラムを実行するには、pubs2 データベースが必要です。

---

usedir.c サンプル・プログラム

usedir.c は、使用できるサーバのリストをディレクトリ・サービスに問い合わせます。

usedir.c は、ドライバ設定ファイル内の定義に従ってデフォルト・ディレクトリで Sybase サーバ・エントリを検索します。ネットワーク・ディレクトリ・サービスが使用されていない場合、*usedir.c* は *sql.ini* ファイルにサーバ・エントリがあるかどうかを調べます。そのあと、検索された各エントリの内容を表示して、接続するサーバをユーザが選択できるようにします。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。ディレクトリ・サービスの詳細については、『Open Client/Server 設定ガイド Windows 版』を参照してください。

exconfig.c サンプル・プログラム

exconfig.c は、Client-Library アプリケーションのプロパティを外部から設定する方法を示します。

このサンプル・プログラムを使用するには、*%SYBASE%¥%SYBASE_OCS%¥ini* 内にあるデフォルト・ランタイム設定ファイル *ocs.cfg* を編集する必要があります。SYBOCS_CFG 環境変数を使用して、*ocs.cfg* ファイルを指すこともできます。

このサンプル・プログラムは、Client-Library プロパティ CS_CONFIG_BY_SERVERNAME を設定し、*server_name* パラメータに "server1" を設定して ct_connect を呼び出します。それに応じて、Client-Library は外部設定ファイルで [server1] セクションを探します。このサンプル・プログラムを実行するには、必要に応じて *ocs.cfg* を編集して、次のセクションを追加してください。

```
[server1]
CS_SERVERNAME = real_server_name
```

real_server_name には、接続するサーバの名前を指定します。

Client-Library での外部設定ファイルの使用方法の詳細については、『Open Client Client-Library/C リファレンス・マニュアル』の「ランタイム設定ファイルの使い方」の項を参照してください。

csr_disp_implicit.c サンプル・プログラム

csr_disp_implicit.c は、暗黙的読み込み専用カーソルの使い方を示します。

- このプログラムは、クエリでカーソルをオープンします。
- 標準の ct_results while ループを使用して結果を処理します。
- カラム値をプログラム変数にバインドします。
- 標準の ct_fetch while ループでローをフェッチして表示します。

プログラムの動作は *csr_disp.c* と同じです。ただし、最初の ct_cursor 呼び出しに、CS_READ_ONLY ではなく CS_IMPLICIT_CURSOR オプションを使用する点だけが異なります。生成される出力は *csr_disp.c* サンプル・プログラムと同じですが、CS_IMPLICIT_CURSOR の使用によりネットワーク・レベルでネットワーク・トラフィックが減少する可能性があります。

このサンプル・プログラムを使用するときは、CS_CURSOR_ROWS オプションに 1 より大きい値を設定します。

クエリは次のとおりです。

```
select au_fname, au_lname, postalcode
from authors
```

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、Adaptive Server バージョン 12.5.1 以降と pubs2 データベースが必要です。

il8n.c サンプル・プログラム

i18n.c は、Client-Library で使用できる次のような国際化機能の一部を示します。

- ローカライズされたエラー・メッセージ
- ユーザ定義のバインド・タイプ

このプログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

マルチスレッドのサンプル・プログラム

このサンプル・プログラムには、マルチスレッド Client-Library アプリケーションの例を示す *multthrd.c* と *thrdfunc.c* という 2 つのファイルが含まれています。

- *multthrd.c* – 5 つのスレッドを生成するソース・コードを含んでいます。各スレッドは 1 つのカーソルまたは 1 つの通常のクエリを処理します。メイン・スレッドはほかのスレッドがクエリ処理を完了するまで待ってから終了します。
- *thrdfunc.c* − サンプル・プログラムが実行に使用するスレッド・ルーチンと同期化ルーチンを決定するプラットフォーム固有の情報を含んでいます。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムは、Client-Library によってサポートされるスレッド・パッケージが、使用しているプラットフォーム上に存在しない場合は実行できません。

csr_disp.c サンプル・プログラム

csr_disp.c は、読み込み専用カーソルの使い方を示します。

- このプログラムは、クエリでカーソルをオープンします。
- 標準の ct_results while ループを使用して結果を処理します。
- カラム値をプログラム変数にバインドします。
- 標準の ct_fetch while ループでローをフェッチして表示します。

クエリは次のとおりです。

```
select au_fname, au_lname, postalcode
from authors
```

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、pubs2 データベースが必要です。

---

uni_csr_disp.c サンプル・プログラム

uni_csr_disp.c は *csr_disp.c* サンプル・プログラムを変更したものです。実行するには unipubs2 データベースが必要です。

- このプログラムは、クエリでカーソルをオープンします。
- 標準の ct_results while ループを使用して結果を処理します。
- カラム値をプログラム変数にバインドします。
- 標準の ct_fetch while ループでローをフェッチして表示します。

クエリは次のとおりです。

```
select au_fname, au_lname, postalcode
from authors
```

unipubs2 データベースをインストールする方法の詳細については、*%SYBASE%¥%SYBASE_OCS%¥sample¥ctlib* に格納されている *README* ファイルを参照してください。

rpc.c コマンドのサンプル・プログラム

リモート・プロシージャ・コール・コマンド RPC のサンプル・プログラム *rpc.c* は、RPC コマンドをサーバに送信してその結果を処理します。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

uni_rpc.c サンプル・プログラム

uni_rpc.c は unichar データ型と univarchar データ型を使用するために *rpc.c* サンプル・プログラムを変更したものです。実行するには unipubs2 データベースが必要です。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

unipubs2 データベースをインストールする方法の詳細については、*%SYBASE%¥%SYBASE_OCS%¥sample¥ctlib* に格納されている *README* ファイルを参照してください。

secct.c サンプル・プログラム

secct.c は、Client-Library アプリケーションでネットワーク・ベースのセキュリティ機能を使用する方法を示します。

このサンプル・プログラムを実行するには、使用するマシンに DCE または Kerberos をインストールして稼働させる必要があります。また、Security Guardian や Open Server のサンプル・プログラム *secsrv.c* などの、ネットワーク・ベースのセキュリティをサポートするサーバに接続することも必要です。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。ネットワーク・セキュリティ・サービスの詳細については、『Open Client/Server 設定ガイド Windows 版』を参照してください。

csr_disp_scrollcurs.c サンプル・プログラム

csr_disp_scrollcurs.c は、スクロール可能カーソルを使用して、pubs2 データベース内の authors テーブルからデータを取り出します。

- クエリをサーバに送信して、カーソルをオープンします。
- 標準の ct_results while ループを使用して結果を処理します。
- カラム値をプログラム変数にバインドします。
- 標準の ct_scroll_fetch while ループでローをフェッチして表示します。

このサンプル・プログラムでは、1 つのプリフェッチ・バッファと、通常のプログラム変数を使用します。クエリは次のとおりです。

```
select au_fname, au_lname, postalcode
from authors
```

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、スクロール可能カーソルをサポートする Adaptive Server バージョン 15.0 以降と pubs2 データベースが必要です。

---

csr_disp_scrollcurs2.c サンプル・プログラム

csr_disp_scrollcurs2.c は、スクロール可能カーソルを使用して、pubs2 データベース内の authors テーブルからデータを取り出します。

- クエリをサーバに送信して、カーソルをオープンします。
- 標準の ct_results while ループを使用して結果を処理します。
- カラム値をプログラム変数にバインドします。
- ct_scroll_fetch を使用してローをフェッチし、表示します。

このサンプル・プログラムは、プログラム変数として配列とともにスクロール可能なカーソルを使用し、配列バインドを使用します。1 回の ct_scroll_fetch 呼び出しの結果が、1 つの配列に表示されます。

クエリは次のとおりです。

```
select au_fname, au_lname, postalcode
from authors
```

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、スクロール可能カーソルをサポートする Adaptive Server バージョン 15.0 以降と pubs2 データベースが必要です。

---

getsend.c サンプル・プログラム

getsend.c は、テキストとその他のデータ型を含んでいるテーブルから text データを取得して更新する方法の例を示すものです。同じプロセスを使用して、image データを取得および更新できます。アプリケーションが Open Server アプリケーションに接続する場合は、その Open Server アプリケーションは Adaptive Server 用の言語コマンドを処理できなければなりません。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意　このサンプル・プログラムを実行するには、pubs2 データベース、authors テーブルが必要です。

---

twophase.c サンプル・プログラム

twophase.c は、2 つの異なるサーバに対して簡単な更新を実行します。このサンプル・プログラムを実行した後で各サーバに対して isql を使用すると、更新が実際に行われたかどうかを調べることができます。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

uni_blktxt.c サンプル・プログラム

uni_blktxt.c は、バルク・コピー・ルーチンを使用して、静的データをサーバ・テーブルにコピーします。このプログラムは、unichar データ型と univarchar データ型を使用します。プログラム変数にバインドされてサーバにまとめて送信される 3 つのローのデータがあります。このローは、テキスト・データを送信するために blk_textxfer を使用してもう一度送信されます。

uni_compute.c サンプル・プログラム

uni_compute.c は unichar データ型と univarchar データ型の計算結果の処理の例を示します。実行するには unipubs2 データベースが必要です。

- クエリを、言語コマンドを使用してサーバに送信します。
- 標準の ct_results while ループを使用して結果を処理します。
- カラム値をプログラム変数にバインドします。
- 標準の ct_fetch while ループでフェッチして表示します。

unipubs2 データベースをインストールする方法の詳細については、*%SYBASE%¥%SYBASE_OCS%¥sample¥ctlib* に格納されている *README* ファイルを参照してください。

wide_compute.c サンプル・プログラム

wide_compute.c は、ワイド・テーブルと大きなカラム・サイズを使用した計算結果の処理を示します。

- クエリを、言語コマンドを使用してサーバに送信します。
- 標準の ct_results while ループを使用して結果を処理します。
- カラム値をプログラム変数にバインドします。
- 標準の ct_fetch while ループでローをフェッチして表示します。

クエリは次のとおりです。

```
select type, price from titles
where type like "%cook"
order by type, price
compute sum(price) by type
compute sum(price)
```

このクエリは、通常のローと計算ローの両方を返します。計算ローは 2 つの compute 句によって生成されます。

- 最初の compute 句は、type の値が変化するたびに計算ローを生成します。

  ```
  compute sum(price) by type
  ```

- 2 つ目の compute 句は、最後に返される 1 つの計算ローを生成します。

  ```
  compute sum(price)
  ```

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

注意 このサンプル・プログラムを実行するには、pubs2 データベースが必要です。

---

wide_curupd.c サンプル・プログラム

wide_curupd.c は、カーソルを使用して pubs2 データベース内の publishers テーブルからデータを取り出します。ローごとにデータを取得し、publishers テーブル内の “state” カラムに新しい値を入力するようユーザに求めるプロンプトを表示します。

update コマンド用の入力パラメータ (“publishers” テーブルの “state” カラム) の値を入力します。次に示すコマンドを実行して publishers3 テーブルを作成してから、サンプル・プログラムを実行してください。

```
use pubs2
go
drop table publishers3
go
create table publishers3 (pub_id char(4) not null,
pub_name varchar(400) null, city varchar(20) null,
state char(2) null)
go
select * into publishers3 from publishers
go
create unique index pubind on publishers3(pub_id)
go
```

wide_dynamic.c サンプル・プログラム

wide_dynamic.c は、カーソルを使用して pubs2 データベース内の publishers テーブルからデータを取り出します。ローごとにデータを取得し、publishers テーブル内の “state” というカラムに新しい値を入力するようユーザに求めるプロンプトを表示します。

このプログラムは、動的 SQL を使用して tempdb データベース内の titles テーブルから値を取り出します。識別子の付いたプレースホルダを含む select 文が、サーバに送信されて部分的にコンパイルされ、保存されます。select を呼び出すたびに、取得されるローを決定するキー値の新しい値だけを渡します。動作は、ストアド・プロシージャに入力パラメータを渡す動作に似ています。また、このプログラムはカーソルを使用してローを 1 つずつ取得します。必要に応じて、この操作を実行できます。

wide_rpc.c サンプル・プログラム

wide_rpc.c は、サーバに RPC コマンドを送信して、結果を処理します。この動作は *wide_rpc.c* プログラムと同じですが、異なる点は、ワイド・テーブルと大きなカラム・サイズを使用することです。

このサンプル・プログラムの詳細については、サンプル・ソース・ファイルの先頭にあるコメントを参照してください。

第 3 章

Open Client DB-Library/C のサンプル・プログラム

Open Client DB-Library はクライアント・アプリケーションの作成に使用できるルーチンの集まりです。DB-Library は、Client-Library 以前の古いルーチンです。ディレクトリ・サービスやセキュリティ・サービスのサポートなどの一部の機能は DB-Library には含まれていません。これらのサービスを利用する場合は Client-Library を使用してください。

DB-Library には、サーバにコマンドを送信するルーチンとそれらのコマンドの結果を処理するルーチンが含まれています。アプリケーション・プロパティの設定、エラー条件の処理、サーバとのアプリケーションの対話に関するさまざまな情報の提供を行うルーチンもあります。

DB-Library のサンプル・プログラムの使用

サンプル・プログラムは、DB-Library/C に固有な機能の例を示します。これらのプログラムは DB-Library/C のトレーニング用ではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

注意 これらの簡単なプログラムは、実際の運用環境で使用するために作成されているものではありません。運用環境で使用するプログラムには、エラー処理や特殊ケースの処理のためのコードを追加する必要があります。

DB-Library アプリケーションの実行に必要な手順は次のとおりです。

- DSQUERY 環境変数を接続先のサーバの名前 (*server_name*) に設定します。
- SYBASE 環境変数を Sybase インストール・ディレクトリのパスに設定していない場合は、設定します。

- SYBASE_OCS 環境変数を、Open Client/Server 製品のホーム・ディレクトリに設定します。たとえば、Open Client/Server バージョン 15.5 のホーム・ディレクトリは、*OCS-15_0* です。
- 付属の *makefile* を使用して make を実行し、*example name* というサンプル実行プログラムを作成します。

その他の条件については、個々のサンプル・プログラムと *README* ファイルを参照してください。

サンプル・プログラムのロケーション

サンプル・プログラムは *%SYBASE%¥%SYBASE_OCS%¥sample¥dblib* にあり、これには以下が含まれています。

- サンプル・プログラムのソース・コード
- サンプル・プログラムのためのデータ・ファイル
- *sybdbex.h* を含むヘッダ・ファイル

注意 サンプル・プログラムが常駐するディレクトリの内容のバックアップ・コピーを作成してください。これによって、元のファイルの整合性に影響を与えずにサンプル・プログラムを使用することができます。

ヘッダ・ファイル

次のヘッダ・ファイルは、すべての DB-Library/C アプリケーションで必要です。

- *sybfront.h* – 関数の戻り値 (『Open Client DB-Library/C リファレンス・マニュアル』 を参照) と、終了値 STDEXIT と ERREXIT などの記号定数を定義しています。*sybfront.h* には、プログラム変数の宣言に使用できるデータ型の型定義も含まれています。
- *sybdb.h* – 追加の定義と型定義を含んでいます。これらの定義のほとんどは、DB-Library/C ルーチンだけで使用されます。*sybdb.h* の内容は、『Open Client DB-Library/C リファレンス・マニュアル』の説明に従って使用してください。
- *syberror.h* – エラー重大度の値を含んでいます。プログラムがこれらの値を参照する場合はインクルードする必要があります。

ヘッダ・ファイルの詳細については、『Open Client DB-Library/C リファレンス・マニュアル』 を参照してください。

sybdbex.h ヘッダ・ファイル

すべてのサンプル・プログラムは、サンプル・ヘッダ・ファイル *sybdbex.h* を参照します。*sybdbex.h* の内容は、次のとおりです。

```
/*
 ** sybdbex.h
 **
 ** This is the header file that goes with the
 ** Sybase DB-Library sample programs.
 **
 **
 */

 #define USER             "sa"
 #define PASSWORD         ""
 #define LANGUAGE         "us_english"
 #define SQLBUFLEN        255
 #define ERR_CH           stderr
 #define OUT_CH           stdout
extern void            error();

int CS_PUBLIC err_handler PROTOTYPE((
DBPROCESS   *dbproc,
int         severity,
int         dberr,
int         oserr,
char       *dberrstr,
char       *oserrstr));

int CS_PUBLIC msg_handler PROTOTYPE((
DBPROCESS   *dbproc,
DBINT       msgno,
int         msgstate,
int         severity,
char        *msgtext,
char        *srvname,
char        *procname,
int          line));
```

データ変換のサンプル・プログラム以外のすべてのサンプル・プログラムには、次の行が含まれています。

```
DBSETLUSER(login, USER);
DBSETLPWD(login, PASSWORD);
```

USER、PASSWORD、LANGUAGE 変数に対し、次の変更を加えます。

- USER は、*sybdbex.h* 内で “sa” と定義されています。サンプル・プログラムを実行する前に *sybdbex.h* を編集して “sa” をサーバのログイン名に置き換えてください。
- PASSWORD は、*sybdbex.h* 内に null (“ ”) 文字列として定義されています。*sybdbex.h* を編集してサーバのパスワードを変更できます。
  - サンプル・プログラムを実行している間だけ、サーバ・パスワードを “” に変更します。ただし、この場合、セキュリティを侵害される可能性があります。つまり、承認されていないユーザでもサーバにログインできます。これでは問題がある場合は、他の 2 つの方法のどちらかを選択してください。
  - *sybdbex.h* 内の “” 文字列を、使用するサーバのパスワードに変更します。オペレーティング・システムの保護メカニズムを使用して、使用中は他のユーザがヘッダ・ファイルにアクセスできないようにします。サンプル・プログラムの使用を終了したら、変更した行を “server_password” に戻します。
  - サンプル・プログラム内で、サーバのパスワードを設定する ct_con_props コードを削除して、ユーザにサーバのパスワードを要求するようにコードを変更します。このコードはプラットフォームに固有なので、Sybase からは提供されません。
- LANGUAGE は、*sybdbex.h* で “us_english” と定義されています。サーバの言語が “us_english” でない場合は、*sybdbex.h* を編集して “us_english” をサーバの言語に変更できます。国際言語ルーチンのサンプル・プログラム *exampl12.c* は、LANGUAGE を参照する唯一のサンプル・プログラムです。

サンプル・プログラムの概要

DB-Library ルーチンの一般的な使い方を示すサンプル・プログラムが提供されています。サンプル・プログラムには、Adaptive Server サンプル・データベースを使用するものもあります。サンプル・データベースをインストールする方法については、『ASE インストール・ガイド』を参照してください。

サンプル・プログラムは C ソース・ファイルです。DB-Library のサンプル・プログラムを使用したり、アプリケーションを構築したりするには、使用するプラットフォームに適切なコンパイラをインストールしてください。

example1.c サンプル・プログラム

example1.c は、1 つのコマンド・バッチで 2 つのクエリを Adaptive Server に送信し、結果をバインドして、返されたデータのローを出力します。

注意　このサンプル・プログラムを実行するには、Adaptive Server にアクセスする必要があります。

example2.c サンプル・プログラム

example2.c は、新しく作成されたテーブルにファイルからデータを挿入し、サーバのローを選択して、結果のバインドと出力を行います。

注意　このサンプル・プログラムを実行するには、Adaptive Server にアクセスする必要があります。また、このサンプル・プログラムには、提供されている *datafile* という名前のファイルと、ログイン・データベースに対する create database パーミッションが必要です。

example3.c サンプル・プログラム

example3.c は、pubs2 データベース内の titles テーブルから情報を選択して出力し、集約結果と計算結果の両方のバインドの例を示します。

注意　このサンプル・プログラムを実行するには、pubs2 データベースを格納している Adaptive Server にアクセスする必要があります。

example4.c サンプル・プログラム

example4.c はロー・バッファリングの例です。このプログラムは、Adaptive Server にクエリを送信し、返されたローをバッファに入れて、それらを対話的に調べることができるようにします。

注意　このサンプル・プログラムを実行するには、Adaptive Server にアクセスする必要があります。

example5.c サンプル・プログラム

example5.c は、データ変換を処理する DB-Library/C ルーチン dbconvert の例を示します。

example6.c サンプル・プログラム

example6.c は、ブラウズ・モードについての例です。このプログラムはテーブルを作成し、そのテーブルにデータを挿入して、ブラウズ・モード・ルーチンを使用してそのテーブルを更新します。ブラウズ・モードはデータのローを一度に 1 つずつ更新するアプリケーションに便利です。

example6.c を使用するには、提供されている *datafile* という名前のファイルが必要です。このプログラムはデフォルト・データベースにテーブル alltypes を作成します。

注意 このサンプル・プログラムを実行するには、Adaptive Server にアクセスする必要があります。

example7.c サンプル・プログラム

example7.c は、ブラウズ・モードを使用してアドホック・クエリによる結果カラムのソースを調べます。

ブラウズ・モードのアプリケーションが更新できるのはブラウズ可能なテーブルから導出されたカラムで、SQL 式の結果ではないカラムだけなので、結果カラムのソースを調べることは重要です。

このサンプル・プログラムは、ブラウズ・モードを使用して更新できる、アドホック・クエリによる結果カラムはどれであるかをアプリケーションが調べる方法を示します。

このサンプル・プログラムは、アドホック・クエリの入力を要求します。select クエリがキーワード for browse を含んでいるかどうか、選択されるテーブルをブラウズできるかどうかによって、結果は異なります。

注意 このサンプル・プログラムを実行するには、Adaptive Server にアクセスする必要があります。

example8.c サンプル・プログラム

example8.c は、RPC (リモート・プロシージャ・コール) を送信し、その RPC による結果ローを表示して、リモート・プロシージャによって返されたパラメータとステータスを表示します。

このサンプル・プログラムを使用するには、デフォルト・データベース内にストアド・プロシージャ rpctest を作成する必要があります。*example8.c* ソース・コードの先頭のコメントは、rpctest を作成するのに必要な create procedure 文を指定します。

注意　このサンプル・プログラムを実行するには、Adaptive Server にアクセスする必要があります。

---

example9.c サンプル・プログラム

example9.c では、ランダムなイメージを生成し、そのイメージをテーブルに挿入してから、挿入されたイメージを選択して元のイメージと比較します。このとき、次の手順に従います。

1　text 値または image 値を除くすべてのデータを、ローに挿入します。

2　ローを更新し、text 値または image 値を NULL に設定します。これは必須の手順です。なぜなら、null である text 値または image 値に有効なテキスト・ポインタが割り当てられるのは、その null 値が update 文によって明示的に入力された場合に限られるからです。

3　ローを選択します。text または image の値が含まれているカラムを明示的に選択してください。これは、アプリケーションの DBPROCESS に正しいテキスト・ポインタとテキスト・タイムスタンプ情報を設定するために必要な手順です。アプリケーションは、この select コマンドによって返されたデータを破棄します。

4　dbtxtptr を呼び出して、テキスト・ポインタを DBPROCESS から取り出します。dbtxtptr の *column* パラメータは整数で、手順 3 で実行された select オペレーションを参照します。たとえば、“text_column” が text カラムの名前である場合、select 文は次のような構文を読み込みます。

```
select date_column, integer_column, text_column
from bigtable
```

dbtxtptr は、*column* が 3 として渡されるように要求します。

5 dbtxtimestamp を呼び出して、DBPROCESS からテキスト・タイムスタンプを取り出します。dbtxtimestamp の *column* パラメータは、手順 3 で実行した select オペレーションを参照します。

6 text 値または image 値を Adaptive Server に書き込みます。アプリケーションは次のどちらかを実行できます。

- 一度の dbwritetext 呼び出しで値を書き込む。
- dbwritetext と dbmoretext を使用して、値をいくつかのまとまりに分けて書き込む。

注意 アプリケーションに、この text 値または image 値に対してさらに更新を実行させるときは、正常に実行された dbwritetext のオペレーションの終りに、Adaptive Server によって返される新しいテキスト・タイムスタンプを保存しなければならない場合があります。新しいテキスト・タイムスタンプには、dbtxtsnewval を使用してアクセスできます。また、あとで取り出せるように dbtxtsput を使用して保存できます。

また、このサンプル・プログラムを実行するには、pubs2 データベースを格納している Adaptive Server にアクセスする必要があります。

---

exampl10.c サンプル・プログラム

exampl10.c は、作家 ID とイメージを含んでいるファイルの名前を要求し、そのファイルからイメージを読み込んで、作家 ID とイメージを含んでいる新しいローを pubs2 データベースのテーブル au_pix に挿入します。詳細については、「example9.c サンプル・プログラム」(37 ページ) を参照してください。

注意 *exampl10.c* を実行するには、Adaptive Server と pubs2 データベースにアクセスする必要があります。著者 ID は “*nnn-nn-nnnn*” の形式 (*n* は数値) でなければなりません。サンプル・コードとともに提供されている *imagefile* には image が入っています。

---

exampl11.c サンプル・プログラム

exampl11.c は、pubs2 データベース内の *au_pix* テーブルからイメージを取り出します。入力する著者 ID によって、プログラムが選択するローが決まります。ローを取り出したあと、このサンプル・プログラムは *pic* カラムに含まれているイメージを指定のファイルにコピーします。

Adaptive Server から text または image 値を取得するには 2 つの方法があります。

- *exampl11.c* では、値を含んでいるローを選択し、dbnextrow を使用してそのローを処理します。dbnextrow が呼び出されると、dbdata を使用して、返されたイメージへのポインタを返すことができます。
- また、dbmoretext と一緒に dbreadtext を使用して、text または image 値をさらに小さないくつかのまとまりとして読み込むこともできます。dbreadtext の詳細については、『Open Client DB-Library/C リファレンス・マニュアル』を参照してください。

注意　このサンプル・プログラムを実行するには、pubs2 データベースを格納している Adaptive Server にアクセスする必要があります。

exampl12.c サンプル・プログラム

exampl12.c は、pubs2 データベースからデータを取り出して us_english フォーマットで出力します。

注意　このサンプル・プログラムを実行するには、pubs2 データベースを格納している Adaptive Server にアクセスする必要があります。

bulkcopy.c サンプル・プログラム

bulkcopy.c は、バルク・コピー・ルーチンを使用して、Adaptive Server の数種のデータ型を含んでいる新しく作成されたテーブルに、ホスト・ファイルからデータをコピーします。

注意　このサンプル・プログラムを実行するには、Adaptive Server にアクセスする必要があります。create database と create table パーミッションを持っていることも必要です。

twophase.c サンプル・プログラム

twophase.c は、2 つの異なるサーバに対して簡単な更新を実行します。実際の更新内容については、ソース・コードを参照してください。このサンプル・プログラムを実行したあとは、各サーバに対して isql を使用して、更新が実際に行われたかどうかを調べることができます。

このサンプル・プログラムでは、“SERVICE” と “PRACTICE” という 2 つの Adaptive Server が稼働していて、それぞれが pubs2 データベースを格納していることを前提としています。使用するサーバの名前がこれと異なる場合は、ソース・コードの “SERVICE” と “PRACTICE” をサーバの実際の名前で置き換えてください。

このサンプル・プログラムを実行する前に、*interfaces* ファイルが両方のサーバについて適切なエントリを持っていることを確認してください。*interfaces* ファイルについては、『Open Client DB-Library/C リファレンス・マニュアル』と『Open Client/Server 設定ガイド Windows 版』を参照してください。

“PRACTICE” サーバが “SERVICE” サーバとは異なるマシン上に存在する場合は、そのマシン上の *interfaces* ファイルも “SERVICE” クエリ・ポート用のエントリを持っている必要があります。

第 4 章

Open Server Server-Library/C のサンプル・プログラム

Open Server Server-Library/C は、クライアント／サーバ・アーキテクチャを利用するサーバを設計するために使用します。これらの Open Server は、Sybase 以外のデータベース管理システムに保管されているデータにアクセスし、外部イベントをトリガし、Open Client アプリケーションに応答します。

クライアント／サーバ・アーキテクチャでは、コンピューティング作業が「クライアント」と「サーバ」間で分担されます。

- クライアントはサーバに要求し、サーバの応答を処理します。
- サーバは要求に応えて、データ、パラメータ、ステータス情報をクライアントに返します。

このアーキテクチャでは、Open Client アプリケーション・プログラムは、Adaptive Server と Open Server によって提供されるサービスを使用するクライアントになります。Server-Library を使用すると、完全なスタンドアロン・サーバを作成できます。

Server-Library のサンプル・プログラムの使用

サンプル・プログラムは、Server-Library/C 固有の機能の例を示しています。これらのプログラムは Server-Library/C のトレーニング用としてではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

注意 これらの簡単なプログラムは、実際の運用環境で使用するために作成されているものではありません。運用環境で使用するプログラムには、エラー処理や特殊ケースの処理のためのコードを追加する必要があります。

Open Server サンプル プログラムを使用する前に、次の操作を実行します。

1 SYBASE 環境変数に Sybase リリース・ディレクトリのパスを設定していない場合は、設定します。

2 SYBASE_OCS 環境変数を、Open Client/Server 製品のホーム・ディレクトリに設定します。たとえば、Open Client/Server バージョン 15.5 のホーム・ディレクトリは、*OCS-15_0* です。

3 DSLISTEN 環境変数に、使用するサーバの名前を設定します。

4 付属の *makefile* を使用して make を実行し、*example_name* というサンプル実行プログラムを作成します。

使用する環境と変数の設定の詳細については、『Open Client/Server 設定ガイド Windows 版』を参照してください。

ロケーションと内容

Server-Library に付属しているサンプル・ファイルは *%SYBASE%¥%SYBASE_OCS%¥sample¥srvlib* にあり、これには以下が含まれています。

- サンプル・プログラムのソース・コード。
- *README* – サンプル・プログラムの構築、実行、トラブルシューティングについて、プラットフォーム固有の一般的な注意が記述されたテキスト・ファイル。
- *makefile* – サンプル・プログラムの構築用に提供されています。Open Server アプリケーションの作成を開始するときにこの *makefile* を使用してください。

- SRV_CONNECT イベント・ハンドラ
- エラー・ハンドラ

注意 サンプル・プログラムが常駐するディレクトリの内容のバックアップ・コピーを作成してください。これによって、元のファイルの整合性に影響を与えずにサンプル・プログラムを使用することができます。

トレース

トレース機能は、アプリケーションによって実行されるアクティビティに関して、選択したオプションに従って詳細な情報を提供します。Open Server のサンプル・プログラムは、トレース機能をサポートしており、トレース出力を Open Server ログ・ファイルに送ります。トレース機能を使用できるようにするには、サンプル・プログラムを実行するときにコマンド・ラインに次のオプションを指定します。

```
example_name [normal_sample_options]
[-h] [-d] [-i] [-a] [-m] [-t] [-e] [-q] [-n]
```

表 4-1 は、各オプションが提供するトレース情報のタイプの説明です。

表 *4-1:* トレース・オプション

オプション	説明
-h	TDS ヘッダ
-d	TDS データ
-i	I/O
-a	アテンション
-m	メッセージ・キュー
-t	TDS トークン
-e	イベント・トレース
-q	遅延イベント・キュー
-n	Net-Library トレース

注意 -e と -q を同時に指定することはできません。

ヘッダ・ファイル

Open Server アプリケーションには、次のヘッダ・ファイルが必要です。

- *ospublic.h* – パブリックな Open Server の構造体、データ型定義、定義文、関数プロトタイプが含まれています。
- *oserror.h* – Open Server のエラー・メッセージの番号とテキストが含まれています。
- *oscompat.h* – 古いデータ型定義、データ型、ルーチン、定数、関数プロトタイプの、新しいバージョンへのマップが含まれています。

『Open Server Server-Library/C リファレンス・マニュアル』を参照してください。

サンプル・プログラムの概要

これらのサンプル・プログラムは、C 言語プログラムでの Server-Library ルーチンの一般的な使用法の例を示すものです。

サンプル・プログラムは C ソース・ファイルです。Server-Library のサンプル・プログラムを使用したり、アプリケーションを構築したりするには、使用するプラットフォーム上に適切なコンパイラをインストールする必要があります。詳細については、「第 1 章 Open Client と Open Server のアプリケーションの構築」を参照してください。

サンプル・プログラムのテスト

サンプル・プログラムを実行する前に、次の操作を行います。

1 リモート・アクセス用に設定されている Adaptive Server にアクセスできるかどうか確認します。これを行うには、Adaptive Server にログインし、以下を入力します。

```
execute sp_configure
```

Adaptive Server がすでにリモート・アクセス用に設定されている場合、“remote access” オプションに対応する config_value および run_value カラムは 1 になっている必要があります。config_value が 0 の場合は、以下を入力します。

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

2 使用する Open Server 名のエントリが *sql.ini* ファイルまたは Windows レジストリ・ファイルに存在していることを確認します。dsedit ユーティリティを使用して、*sql.ini* ファイル または Windows レジストリ・ファイルにエントリを作成します。dsedit の詳細については、『Open Client/Server 設定ガイド Windows 版』を参照してください)。

3　使用する Open Server 名のエントリが Adaptive Server の sysservers テーブルに存在していることを確認します。これを確認するには、Adaptive Server にログインし、以下を入力します。

```
execute sp_helpserver
```

このコマンドは、Adaptive Server の sysservers テーブルの中で使用可能なすべてのサーバのリストを返します。使用する Open Server の名前がこのリストにない場合は、以下を入力します。

```
execute sp_addserver your_open_server_name
```

4　以下の環境変数がまだ設定されていない場合は、設定します。

環境変数	値
SYBASE	Sybase ホーム・ディレクトリのロケーション。
DSLISTEN	*sql.ini* ファイルまたはディレクトリ・サービスにリストされている Open Server アプリケーションの名前。
DSQUERY	*sql.ini* ファイルまたはディレクトリ・サービスにリストされている Open Server の名前。

osintro.c サンプル・プログラム

osintro.c は、Open Server アプリケーションを構築するための基本的なコンポーネントの例を示すものです。*osintro.c* には言語ハンドラは含まれていないので、isql コマンドを読み込むことはできません。

ctos.c サンプル・プログラム

ctos.c は、ゲートウェイ Open Server のアプリケーションです。このプログラムは、Server-Library 呼び出しと Client-Library 呼び出しを使用します。まず、クライアントからコマンドを受け取って、リモート Adaptive Server に渡し、次にリモート・サーバから結果を取り出して、クライアントに渡します。*ctos.c* は、次のさまざまなクライアント・コマンドを処理します。

- バルク・コピー・コマンド
- カーソル・コマンド
- スクロール可能カーソル・コマンド
- 動的 SQL コマンド
- 言語コマンド
- メッセージ・コマンド
- オプション・コマンド
- RPC (リモート・プロシージャ・コール)

ctos.c はさらに、SRV_ATTENTION イベント・ハンドラを呼び出すことによってクライアントからのアテンション要求に応答します。このプログラムには、各タイプのクライアント・コマンドを処理するためにイベント・ハンドラ・ルーチンが含まれています。

注意 Server-Library に付属している他のサンプル・プログラムと違って、*ctos.c* は完成品に近いものです。このサンプル・プログラムは、運用環境で使用できるコーディング・テンプレートとして提供されています。*ctos.c* プログラムを終了するには、コマンド・ウィンドウから [Ctrl + C] キーを押します。*README* ファイル内のコマンドは間違っています。

ゲートウェイの詳細については、『Open Server Server-Library/C リファレンス・マニュアル』を参照してください。

lang.c サンプル・プログラム

lang.c は、SRV_LANGUAGE イベント・ハンドラの使用例を示すものです。このイベント・ハンドラは、情報メッセージを使用してクライアントの言語コマンドに応答します。このとき、イベント・ハンドラは srv_sendinfo ルーチンを使用して情報メッセージをクライアントに送信します。このプログラムには、SRV_CONNECT イベント・ハンドラとエラー・ハンドラも含まれています。

詳細については、『Open Server Server-Library/C リファレンス・マニュアル』の「言語呼び出し」を参照してください。

fullpass.c サンプル・プログラム

fullpass.c は、TDS (Tabular Data Stream™) パススルー・モードを使用する Open Server ゲートウェイ・アプリケーションです。詳細については、『Open Server Server-Library/C リファレンス・マニュアル』の「パススルー・モード」を参照してください。

イベント・ハンドラ・ルーチンは srv_recvpassthru を使用してクライアント要求を受け取り、ct_sendpassthru ルーチンを使用してこの情報を Adaptive Server に転送します。クライアント・コマンド全体がリモート・サーバに転送されると、イベント・ハンドラは ct_recvpassthru を使用してリモート・サーバから結果を読み込み、srv_sendpassthru を使用してクライアントに渡します。

このアプリケーションには、SRV_CONNECT イベント・ハンドラも含まれています。このハンドラは、srv_getloginfo と ct_setloginfo を使用して、クライアント接続情報をリモート・サーバに転送します。次に ct_getloginfo と srv_setloginfo を使用して、接続確認情報をクライアントに渡します。TDS パススルー・モードを使用するすべての Open Server アプリケーションは、その SRV_CONNECT イベント・ハンドラ内にこれらの呼び出しを含んでいる必要があります。

注意　このアプリケーションを実行するには、Adaptive Server にアクセスする必要があります。

regproc.c サンプル・プログラム

regproc.c は、Open Server 11.1 以降でのレジスタード・プロシージャの使用例を示すものです。このアプリケーションは起動時にいくつかのプロシージャを登録してからクライアントのコマンドを待ちます。Open Server イベント・ハンドラはインストールされません。

クライアントは RPC コマンドを送信して、*regproc.c* に定義されているレジスタード・プロシージャを実行します。

regproc.c で使用するためのクライアント・プログラムが、次のようにいくつか追加されています。

- *version.c* － Open Server のバージョン番号をクライアントに返すレジスタード・プロシージャ (rp_version) を実行します。
- *dbwait.c* － DB-Library で実装されており、レジスタード・プロシージャ rp_version が実行されるときにクライアントに通知するように Open Server に要求します。
- *ctwait.c* － Client-Library で実装されており、レジスタード・プロシージャ rp_version が実行されるときにクライアントに通知するように Open Server に要求します。

intlchar.c サンプル・プログラム

intlchar.c は、Open Server が国際言語と文字セットを処理する方法の例を示します。このサンプル・プログラムは、Open Server アプリケーションのネイティブ言語と文字セット用の値を初期設定して、クライアント要求に応答してこれらの値を変更します。

クライアント要求は、オプション・コマンドと言語コマンドのフォーマットで渡されます。*intlchar.c* は、SRV_OPTION イベント・ハンドラと SRV_LANGUAGE イベント・ハンドラ、および SRV_CONNECT ハンドラをインストールします。

multthrd.c サンプル・プログラム

multthrd.c は、次のようないくつかの Open Server マルチスレッド・プログラミング機能の例を示します。

- srv_spawn によるサービス・スレッドの作成
- クライアント接続スレッドとサービス・スレッド間でのメッセージ・キューによるスレッド間通信 (srv_getmsgq と srv_putmsgq を使用)
- スリープ・メカニズムとウェイクアップ・メカニズム (srv_sleep と srv_wakeup を使用)
- スケジューリング情報をレポートするためのコールバック・ルーチンの使用 (srv_callback を使用)

multthrd.c は、SRV_START ハンドラ、SRV_LANGUAGE ハンドラ、SRV_CONNECT ハンドラ、コールバック・ハンドラをインストールします。サービス・スレッドは、Open Server アプリケーションが受け取るすべての言語クエリのログを取ります。実行される内容は次のとおりです。

- アプリケーションの言語ハンドラでは、クライアント・スレッドはクライアントからクエリを読み込み、メッセージ・データとしてクエリを使用してメッセージを「ロガー」というサービス・スレッドに送信します。
- 送信後、クライアント・スレッドは待機します (srv_sleep)。サービス・スレッドは、メッセージを受け取るとクライアント・スレッドをウェイクアップします (srv_wakeup)。
- logger は、継続的にループしてメッセージを待ちます。メッセージを受け取ると、logger はクエリの内容をファイルに書き込んで送信側に通知します。
- ロガーとクライアント・スレッドは、SRV_C_RESUME、SRV_C_SUSPEND、SRV_C_TIMESLICE、SRV_C_EXIT コールバック・ハンドラをインストールして、スケジュール情報を出力します。

secsrv.c サンプル・プログラム

secsrv.c は、Open Server のネットワーク・ベースのセキュリティ・サービスの使用例を示します。

このサンプル・プログラム内の接続ハンドラは、クライアント・スレッドのセキュリティ・プロパティを取り出し、そのセッションでどのセキュリティ・サービスがアクティブになっているかを示すメッセージをクライアントに送信します。

セキュリティ・サービスの詳細については、『Open Client/Server 設定ガイド Windows 版』を参照してください。

第 5 章

Open Client Embedded SQL/C

Embedded SQL は、C などの言語で作成されたアプリケーション・プログラム内に Transact-SQL 文を埋め込むための Transact-SQL™ のスーパーセットです。Embedded SQL には、すべての Transact-SQL 文に加えて、アプリケーション・プログラムで Transact-SQL を使用するために必要な拡張機能が含まれています。

Embedded SQL は、Adaptive Server データベースに保管されているデータの検索、挿入、修正を行うための方法を提供します。

Embedded SQL/C 実行プログラムの構築

Embedded SQL アプリケーションから実行プログラムを構築するには、次の手順に従います。

1 アプリケーションをプリコンパイルします。

2 プリコンパイラによって生成された C ソース・コードをコンパイルします。

3 アプリケーションを必要に応じてオブジェクトやライブラリとリンクします。

4 プリコンパイラによって生成されたストアド・プロシージャをロードします。

アプリケーションのプリコンパイル

以下を使用して、Embedded SQL/C コードをプリコンパイルしてから、アプリケーションをコンパイルしてください。

```
cpre
    [-Ccompiler]
    [-Ddatabase_name]
    [-Ffips_level]
    [-G[isql_file_name]]
    [-H]
    [-Iinclude_path_name]
    [-Jcharset_locale_name]
    [-Ksyntax_level]
    [-L[listing_file_name]]
    [-Ninterface_file_name]
    [-Otarget_file_name]
    [-Ppassword]
    [-Sserver_name]
    [-Ttag_id]
    [-Uuser_id]
    [-Vversion_number]
    [-Zlanguage_locale_name]
    [@options_file]...
   [-a] [-b] [-c] [-d] [-e] [-f] [-h] [-l] [-m] [-p] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
   filename[.ext]
```

注意 スラッシュ (/) またはハイフン (-) のどちらを使用してもオプションをフラグできます。したがって、cpre /I と cpre -I は同じことを表します。

正しくないオプションを指定した場合は、プリコンパイラは使用可能なオプションのリストを表示します。

filename には、Embedded SQL/C ソース・ファイルの名前を指定します。*filename* のデフォルトの拡張子は *.cp* です。cpre 文を使用すると、拡張子が *.c* の出力ファイルが生成されます。

一部のオプションは、ストアド・プロシージャの生成などの、プリコンパイラの機能を有効にします。デフォルトでは、これらの機能は無効になっています。これらの機能を有効にするには、cpre コマンド・ラインでオプションを指定します。このほかの文修飾子は、パスワードなど、プリプロセッサに対する値を指定します。値はオプションのあとに入力します (間にスペースを入れても入れなくてもかまいません)。

cpre オプションの詳細については、「付録 A ユーティリティ・コマンド・リファレンス」を参照してください。

アプリケーションのコンパイルとリンク

Embedded SQL バージョン 11.1 以降は、Windows プラットフォームの Microsoft C コンパイラを使用して動作確認されています。コンパイルとリンク用の実際の構文については *makefile* を参照してください。*makefile* は *%SYBASE%¥%SYBASE_OCS%¥sample¥esqlc* にあります。

リンク・ライブラリ

リンク行には次のライブラリを指定する必要があります。

- *libsybct* － Client-Library DLL
- *libsybcs* － CS-Library DLL
- *libsybtcl* － トランスポート制御層 DLL
- *libsybcomn* － 内部共通ライブラリ DLL
- *libsybintl* － ローカライゼーション・サポート・ライブラリ DLL
- *libsybunic* － Unicode ライブラリ DLL

ストアド・プロシージャのロード

Embedded SQL/C プログラムを実行する前に、プリコンパイラで生成されたストアド・プロシージャを Adaptive Server にロードします。このためには、-G オプションを使用してプログラムをプリコンパイルします。-G オプションは isql スクリプト・ファイルを作成します。次に、isql -i オプションを使用して、作成されたファイルをロードします。

isql の詳細については、「付録 A ユーティリティ・コマンド・リファレンス」を参照してください。

Embedded SQL/C サンプル・プログラムの使用

Embedded SQL には、一般的な Embedded SQL アプリケーションの例を示す 2 つのサンプル・プログラムが付属します。これらは、*%SYBASE%¥%SYBASE_OCS%¥sample¥esqlc* にあります。このディレクトリには、*README* ファイルと *makefile* も含まれています。この項では、これらのサンプル・プログラムの概要について説明します。

サンプル・プログラムは、Embedded SQL/C に固有の機能の例を示しています。これらのプログラムは Embedded SQL/C のトレーニング用ではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

注意 これらの簡単なプログラムは、実際の運用環境で使用するために作成されているものではありません。運用環境で使用するプログラムには、エラー処理のためのコードを追加する必要があります。

Embedded SQL/C のサンプル・プログラムを実行するには、以下の環境変数が設定されていることを確認する必要があります。

- SYBASE – Sybase インストール・ディレクトリのパスに設定します。
- SYBASE_OCS – Open Client/Server 製品のホーム・ディレクトリに設定します。たとえば、Open Client/Server バージョン 15.5 製品のホーム・ディレクトリは、*OCS-15_0* です。

sql.ini ファイルを調べて、使用されるサーバ名のエントリが存在することを確認します。*sql.ini* ファイルを調べるには、dsedit を使用します。『Open Client/Server 設定ガイド Windows 版』で説明しているように、*sql.ini* ファイルにサーバを追加した場合は、ocscfg を使用して各サーバへの接続をテストできます。

サンプル・プログラムを実行するには、pubs2 データベースを格納している Adaptive Server にアクセスする必要があります。pubs2 データベースをインストールする方法については、『ASE インストール・ガイド』を参照してください。

サンプル・プログラムをプリコンパイルする前に、次に示すようにサンプル・ヘッダ・ファイルを編集し、ユーザ名とパスワードを Adaptive Server で有効な値に置き換えておく必要があります。変更箇所についてはプログラム内のコメントを参照してください。

ヘッダ・ファイル

すべてのサンプル・プログラムは、サンプル・ヘッダ・ファイル *sybsqlex.h* を参照します。*sybsqlex.h* の内容は、次のとおりです。

```
/**************************************************
 *                                                *
 *  sybsqlex.h - header file for Embedded SQL/C   *
 *                 sample programs                *
 *                                                *
 **************************************************/
```

```
#define USER      "username"

#define PASSWORD  "password"


#define ERREXIT     -1
#define STDEXIT     0
```

すべてのサンプル・プログラムには、次の行が含まれています。

```
#include "sybsqlex.h"
```

USER および PASSWORD は、*sybsqlex.h* でそれぞれ “username” および “password” として定義されています。サンプル・プログラムを実行する前に *sybsqlex.h* を編集する必要があります。“username” を Adaptive Server のログイン名に、“password” を Adaptive Server のパスワードに変更してください。

example1.cp サンプル・プログラム

example1.cp は、対話型クエリ・プログラムでの通常の非スクロール可能カーソルの使い方を示します。*example4.pc* と *example5.pc* は、スクロール可能カーソルの使い方を示します。これら 3 つのプログラムすべてが、次のように動作します。

- 本のタイプのリストを表示します。ユーザはタイプを 1 つ選択します。
- 選択されたタイプの本のすべてのタイトルを表示し、タイトル ID を要求します。
- 選択されたタイトルについての詳細情報を表示し、さらにタイトル ID を要求します。
- プロンプト画面で [Return] キーが押されると終了します。

example2.cp サンプル・プログラム

example2.cp は、カーソルを使用してローを更新する方法を示しています。このプログラムは次のように動作します。

- 著者テーブル内のカラムをローごとに表示します。
- ユーザは au_id カラムを除くすべてのカラム内の作家情報を更新できます。ユーザがカラム情報に対して [Return] キーを押した場合は、そのカラムのデータは変更されないでもとのままになります。
- ユーザが更新を確認すると、データを Adaptive Server に送信します。

exampleHA.cp サンプル・プログラム

exampleHA.cp は、高可用性 (HA) フェールオーバ機能とともに Embedded SQL/C を使用する方法を示します。このプログラムは、*example1.cp* にフェールオーバ処理が追加されたプログラムと考えられます。エラー・ハンドラが、フェールオーバを検出および処理します。

uni_example1.cp サンプル・プログラム

uni_example1.cp は、titles テーブルの対話型クエリを行うときのカーソルの使い方を示します。このプログラムは、*example1.cp* に unichar/univarchar カラムの表示処理が追加されたプログラムと考えられます。このプログラムは次のように動作します。

- character データ型を unichar/univarchar カラムにバインドします。
- サーバから unichar/univarchar データにアクセスして、クライアントの文字セットの文字フォーマットで表示します。

uni_example2.cp サンプル・プログラム

uni_example2.cp は、テーブルのローの表示と編集を行うときのカーソルの使い方を示します。このプログラムは、*example2.cp* に unichar/univarchar カラムの表示処理が追加されたプログラムと考えられます。このプログラムは次のように動作します。

- character データ型を unichar/univarchar カラムにバインドします。
- サーバから unichar/univarchar データにアクセスして、クライアントの文字セットの文字フォーマットで表示します。

第 6 章

Open Client Embedded SQL/COBOL

Embedded SQL は Transact-SQL のスーパーセットであり、COBOL 言語などで作成されるアプリケーション・プログラムに Transact-SQL 文を埋め込むことができます。Embedded SQL には、すべての Transact-SQL 文に加えて、アプリケーション・プログラムで Transact-SQL を使用するために必要な拡張機能が含まれています。

Embedded SQL/COBOL は、Adaptive Server データベースに保管されているデータの検索、挿入、修正を行うための簡単な方法を提供します。

Embedded SQL/COBOL 実行プログラムの構築

Embedded SQL アプリケーションから実行プログラムを構築するには、次の手順に従います。

1 アプリケーションをプリコンパイルします。

2 プリコンパイラによって生成された COBOL ソース・コードをコンパイルします。

3 アプリケーションを必要に応じてファイルやライブラリとリンクします。

4 プリコンパイラによって生成されたストアド・プロシージャをロードします。

アプリケーションのプリコンパイル

Embedded SQL ソース・プログラムをプリコンパイルするための構文は、次のとおりです。

```
cobpre
    [-Ccompiler]
    [-Ddatabase_name]
    [-Ffips_level]
    [-G[isql_file_name]]
    [-Iinclude_path_name]
    [-Jcharset_locale_name]
    [-Ksyntax_level]
    [-L[listing_file_name]]
    [-Ninterface_file_name]
    [-Otarget_file_name]
    [-Ppassword]
    [-Sserver_name]
    [-Ttag_id]
    [-Uuser_id]
    [-Vversion_number]
    [-Zlanguage_locale_name]
    [@ options_file]
    [-a] [-b] [-c] [-d] [-e] [-f] [-l] [-m] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
    filename[.ext]
```

注意 スラッシュ (/) またはハイフン (-) のどちらを使用しても、オプションをフラグできます。したがって、cobpre -I と cobpre /I は同じことを表します。

正しくないオプションを指定した場合は、プリコンパイラは使用可能なオプションのリストを表示します。

filename は、Embedded SQL/COBOL ソース・ファイルの名前です。ソース・ファイル名の指定は必須です。*filename* のデフォルトの拡張子は *.pco* です。cobpre オプションを指定すると、拡張子が *.cbl* の出力ファイルが生成されます。

一部のオプションは、ストアド・プロシージャの生成などの、プリコンパイラの機能を有効にします。デフォルトでは、これらの機能は「オフ」になっています。これらの機能をオンにするには、cobpre 文の行にオプションを指定します。このほかの文修飾子は、パスワードなど、プリプロセッサに対する値を指定します。値はオプションのあとに入力します (間にスペースを入れても入れなくてもかまいません)。

cobpre のオプションの詳細については、「付録 A ユーティリティ・コマンド・リファレンス」を参照してください。

アプリケーションのコンパイルとリンク

Embedded SQL バージョンは、Micro Focus Net Express 5.1 で動作確認されています。コンパイルとリンク用の実際の構文については、*%SYBASE%¥%SYBASE_OCS%¥sample¥esqlcob* にある *makefile* を参照してください。

リンク・ライブラリ

リンク・コマンド行には次のライブラリの一部または全部を指定する必要があります。

- *libsybcobct* － Client-Library への COBOL インタフェース
- *libsybct* － Client-Library DLL
- *libsybcs* － CS-Library DLL
- *libsybtcl* － トランスポート制御層 DLL
- *libsybcomn* － 内部共通ライブラリ DLL
- *libsybintl* － ローカライゼーション・サポート・ライブラリ DLL
- *libsybunic* － Unicode ライブラリ DLL

ストアド・プロシージャのロード

プリコンパイル時に -G オプションを使用した場合は、プリコンパイラで生成されたストアド・プロシージャを Adaptive Server にロードします。この作業は、isql -i オプションを使用して実行できます。

isql の詳細については、「付録 A ユーティリティ・コマンド・リファレンス」を参照してください。

Embedded SQL/COBOL のサンプル・プログラムの使用

Embedded SQL には、一般的な Embedded SQL アプリケーションの例を示すサンプル・プログラムが含まれています。サンプル・プログラムは *%SYBASE%¥%SYBASE_OCS%¥sample¥esqlcob* に格納されています。

同じディレクトリ内の *README* ファイルには、サンプル・プログラムを構築して実行するための手順とそれらを使用するときの注意が記載されています。*COBOL.pco* ファイルは、Adaptive Server のログイン名とパスワードを定義します。サンプル・プログラムをコンパイルする前に、このファイルの中にあるログイン情報を更新してください。

サンプル・プログラムは、Embedded SQL/COBOL に固有の機能の例を示しています。これらのプログラムは Embedded SQL/COBOL のトレーニング用ではなく、アプリケーション・プログラマのためのガイドとして設計されています。サンプル・プログラムを使用する前に、各ソース・ファイルの先頭にある説明を読んで、ソース・コードの内容を確認してください。

注意 これらの簡単なプログラムは、実際の運用環境で使用するために作成されているものではありません。運用環境で使用するプログラムには、エラー処理のためのコードを追加する必要があります。

Embedded SQL/COBOL のサンプル・プログラムを実行するには、以下の環境変数が設定されているのを確認する必要があります。

- SYBASE – Sybase インストール・ディレクトリのパスに設定していない場合は、設定します。
- SYBASE_OCS – Open Client/Server 製品のホーム・ディレクトリに設定します。たとえば、Open Client/Server バージョン 15.5 製品のホーム・ディレクトリは、*OCS-15_0* です。

pubs2 サンプル・データベースがインストールされている Adaptive Server にアクセスできる必要があります。pubs2 データベースをインストールする方法については、Adaptive Server Enterprise の『インストール・ガイド』を参照してください。

プログラムをプリコンパイルする前に、ユーザ名とパスワードを Adaptive Server で有効な値に置き換えてください。変更箇所についてはプログラム内のコメントを参照してください。

注意 サンプル・プログラムの結果を表示するには、[Return] キーを押す必要があります。

Micro Focus COBOL 用の環境変数

Embedded SQL/COBOL のサンプル・プログラムを実行する前に、表 6-1 に示す環境変数を設定してください。

表 *6-1: COBOL* コンパイラ用の環境変数

環境変数	値
COBDIR	COBOL コンパイラのインストール・ディレクトリの絶対パス
COBLIB	*%COBDIR%¥lib*
PATH	*%COBDIR%¥bin;%COBDIR%¥lib*
LIB	*%COBDIR%¥lib;%LIB%*

example1.pco サンプル・プログラム

example1.pco は、対話型クエリ・プログラムでの通常の非スクロール可能カーソルの使い方を示します。通常の非スクロール可能カーソルのサンプル・プログラムは次のように動作します。

- 本のタイプのリストを表示します。ユーザはタイプを 1 つ選択します。
- 選択されたタイプの本のすべてのタイトルを表示し、タイトル ID を要求します。
- 選択されたタイトルについての詳細情報を表示し、さらにタイトル ID を要求します。
- プロンプト画面で [Return] キーが押されると終了します。

スクロール可能カーソルのサンプル・プログラムは、*example3.pco* と *example4.pco* です。スクロール可能カーソルのサンプル・プログラムは次のように動作します。

- スクロール可能カーソル INSENSITIVE または SEMI_SENSITIVE を宣言します。
- 方向やオフセットとともにハードコードされた FETCHES に基づいて、本のタイトルの選択内容を表示します。
- 処理が完了したら終了します。

example2.pco サンプル・プログラム

example2.pco は、カーソルを使用してローを更新する方法を示しています。このプログラムは次のように動作します。

- 著者テーブル内のカラムをローごとに表示します。
- ユーザは au_id カラムを除くすべてのカラム内の作家情報を更新できます。ユーザがカラム情報に対して [Return] キーを押した場合は、そのカラムのデータは変更されないでもとのままになります。
- ユーザが更新を確認すると、データを Adaptive Server に送信します。

付 録　A

ユーティリティ・コマンド・リファレンス

この付録では、ユーティリティ・プログラムについて説明します。

ユーティリティ	説明	ページ
bcp	ユーザ指定のフォーマットで、データベース・テーブルをオペレーティング・システム・ファイルに、またはオペレーティング・システム・ファイルをデータベース・テーブルにコピーする、バルク・コピー・ユーティリティです。	62
cpre	C ソース・プログラムをプリコンパイルして、ターゲット・ファイル、リスティング・ファイル、isql ファイルを生成する、C プリコンパイラ・ユーティリティです。	83
cobpre	COBOL ソース・プログラムをプリコンパイルして、ターゲット・ファイル、リスティング・ファイル、isql ファイルを生成する、COBOL プリコンパイラ・ユーティリティです。	93
defncopy	指定されたビュー、ルール、デフォルト、トリガ、プロシージャ、レポートの定義をデータベースからオペレーティング・システム・ファイルに、またはオペレーティング・システム・ファイルからデータベースにコピーする、定義コピー・ユーティリティです。	102
isql	Adaptive Server または Open Server に接続してクエリを実行する、対話型 SQL パーサです。	107
instjava	JAR をクライアント・ファイルから Adaptive Server にインストールする、インストール Java ユーティリティです。	126
extrjava	保持されている JAR とそこに含まれるクラスを Adaptive Server からクライアント・ファイルにコピーする、抽出 Java ユーティリティです。	130

bcp

説明

ユーザが指定したフォーマットで、データベース・テーブルをオペレーティング・システム・ファイルに、またはオペレーティング・システム・ファイルをデータベース・テーブルにコピーします。bcp は、*%SYBASE%¥%SYBASE_OCS%¥bin* にあります。

構文

bcp [[*database_name*.]*owner*.]*table_name* [:*slice_number* | partition *partition_name*] {in | out} *datafile*
[-a *display_charset*]
[-A *packet_size*]
[-b *batch_size*]
[-c]
[-C]
[-d *discardfileprefix*]
[-e *errfile*]
[-E]
[-f *formatfile*]
[-F *firstrow*]
[-g *id_start_value*]
[-i *input_file*]
[-I *interfaces_file*]
[-J *client_character_set*]
[-K *keytab_file*]
[-L *lastrow*]
[-m *maxerrors*]
[-n]
[-N]
[-o *output file*]
[-P *password*]
[-Q]
[-r *row_terminator*]
[-R *remote_server_principal*]
[-S *server*]
[-t *field_terminator*]
[-T *text_or_image_size*]
[-U *username*]
[-v]
[-V [*security_options*]]
[-W]
[-X]
[-y *alternate_home_directory*]
[-Y]
[-z *language*]
[-Z *security_mechanism*]
[--colpasswd [[[db_name.[owner].]table_name.]
column_name [*password*]]]
[--hide-vcc]
[--initstring "*TSQL_command*"]
[--keypasswd [[db_name.[owner].]*key_name* [*password*]]]
[--maxconn *maximum_connections*]
[--show-fi]
[--skiprows *nSkipRows*]

パラメータ

database_name

コピーするテーブルがデフォルト・データベースまたは master データベースにある場合、このパラメータはオプションです。それ以外の場合は、データベース名を指定する必要があります。

owner

コピーするテーブルをユーザまたはデータベース所有者が所有している場合、このパラメータはオプションです。所有者を指定しない場合、bcp は、まずユーザが所有するこの名前のテーブルを探します。次に、データベース所有者が所有するテーブルを探します。別のユーザがテーブルを所有している場合は、所有者の名前を指定する必要があります。指定しないと、コマンドは失敗します。

table_name

コピーするデータベース・テーブルの名前です。Transact-SQL の予約語をテーブル名に使用することはできません。

slice_number

コピーするデータベース・テーブルのスライスの番号です。

partition *partition_name*

Adaptive Server のパーティションの名前です。パーティションが複数ある場合は、カンマで区切ったリストを使用します。

in | out

コピーの方向を示します。in はファイルからデータベース・テーブルへのコピーであることを示し、out はデータベース・テーブルからファイルへのコピーであることを示します。

注意　コピー・インまたはコピー・アウトするローの数が 2147483647 を超えた場合、bcp はエラーを発生させ、オペレーションを停止します。

---

datafile

オペレーティング・システム・ファイルのフル・パス名です。パス名は、1 ～ 255 文字で指定します。複数のファイルをリストする場合は、カンマで区切ったリストを使用します。データ・ファイルとパーティションの数が同じであることが必要です。

-a *display_charset*

bcp を実行しているマシンの文字セットと異なる文字セットを使用する端末から、bcp を実行できます。-a を -J とともに使用して、変換に必要な文字セット変換ファイル (*.xlt* ファイル) を指定します。-J を指定しないで -a を使用するのは、クライアントの文字セットがデフォルトの文字セットと同じである場合だけです。

-a パラメータで指定した文字変換ファイルが見つからない場合、または入力したファイル名に誤りがある場合は、次のエラー・メッセージが表示されます。

```
Error in attempting to determine the size of a pair of
translation tables. : 'stat' utility failed.
```

-A *packet_size*

この bcp セッションで使用するネットワーク・パケット・サイズを指定します。たとえば、この bcp セッションのパケット・サイズを 4096 バイトに設定するには、次のように入力します。

```
bcp pubs2..titles out table_out -A 4096
```

packet_size は、default network packet size 設定変数と maximum network packet size 設定変数の間の値であり、512 の倍数であることが必要です。

大規模なバルク・コピー・オペレーションのパフォーマンスを向上させるには、デフォルトよりも大きいネットワーク・パケット・サイズを使用します。

-b *batchsize*

バッチごとにコピーされるデータのロー数です。デフォルトでは、bcp in は 1 つのバッチ処理で *n* 個のローをコピーします。*n* はバッチ・サイズに相当します。バッチ・サイズが適用されるのは、バルク・コピー・インの場合だけです。バルク・コピー・アウトには影響しません。*batchsize* に使用できる最小値は 1 です。

注意 バッチ・サイズを 1 に設定すると、Adaptive Server はコピー・インする 1 つのローに 1 つのデータ・ページを割り付けます。このパラメータは、高速 bcp にのみ適用され、データの破損したローを見つける場合に役立ちます。-b 1 は慎重に使用してください。これを使用すると、ローごとに新しいページが割り付けられるため、領域の使用効率が低下します。

-c

データ・ファイル内の全カラムのデフォルト記憶タイプとして char データ型を使用して、コピー・オペレーションを実行します。プラットフォーム間でデータを共有するときにこのフォーマットを使用してください。このパラメータは、各フィールドのプロンプトを表示しません。また、デフォルトの記憶タイプとして char を使用し、プレフィクスは付きません。デフォルトのフィールド・ターミネータとして ¥t (タブ) を使用し、デフォルトのロー・ターミネータとして ¥n (改行) を使用します。

-C

Adaptive Server が暗号化カラムをサポートしている場合は、暗号化カラムのバルク・コピーをサポートします。-C を指定すると、ciphertext オプションを有効にしてから、バルク・コピー・オペレーションが開始されます。

-d *discardfileprefix*

拒否されたローを専用の破棄ファイルに記録します。破棄ファイルのフォーマットはホスト・ファイルと同じです。このファイルは、指定された破棄ファイル・プレフィクスの後に入力ファイル名を追加することによって作成されます。このファイル内のローを修正し、それを使用して修正後のローを再ロードできます。

破棄ファイルに記録された問題のあるローを特定し、診断するために、-e *errorfile* とともに -d *discardfileprefix* オプションを使用することをおすすめします。

-e *errfile*

bcp がファイルからデータベースに転送できなかったすべてのローを保管するエラー・ファイルのフル・パス名です。bcp プログラムからのエラー・メッセージは、使用している端末に表示され、エラー・ファイルにも記録されます。bcp がエラー・ファイルを作成するのは、このパラメータを指定した場合だけです。複数のセッションが使用されている場合は、エラーのパーティション情報とファイル名情報がエラー・ファイルに追加されます。

破棄ファイルに記録された問題のあるローを特定し、診断するために、-d *discardfileprefix* とともに -e *errorfile* オプションを使用することをおすすめします。

-E

テーブルの IDENTITY カラムの値を明示的に指定します。

デフォルトでは、IDENTITY カラムがあるテーブルにデータをバルク・コピーするときに、bcp は IDENTITY カラムのテンポラリの値 0 を各ローに割り当てます。これは、テーブルにデータをコピーする場合にだけ有効です。bcp はデータ・ファイルから ID カラムの値を読み込みますが、この値をサーバには送信しません。代わりに、bcp がテーブルに各ローを挿入すると、サーバが値 1 で始まる連続したユニークな IDENTITY カラム値をローに割り当てます。データをテーブルにコピーするときに -E フラグを指定した場合は、bcp はデータ・ファイルから値を読み込み、値をテーブルに挿入するサーバに送信します。挿入されるローの数が使用可能な IDENTITY カラム値の最大値を超える場合、Adaptive Server はエラーを返します。

デフォルトでは、IDENTITY カラムを持つテーブルからデータをバルク・コピーすると、bcp はカラムに関するすべての情報を出力ファイルから除外します。-E フラグを指定すると、bcp は既存の IDENTITY カラム値を出力ファイルにコピーします。

-E パラメータは、バルク・コピー・アウトには影響しません。-N パラメータを使用しない場合、Adaptive Server は ID カラムをデータ・ファイルにコピーします。

-E フラグと -g フラグを同時に使用することはできません。

-f *formatfile*

同じテーブルでの前回の bcp 実行時の応答が保管されているファイルのフル・パス名です。bcp がフォーマットについてたずねてきたときに、使用するフォーマットを指定すると、そのフォーマットをフォーマット・ファイルに保存できます。フォーマット・ファイルの作成はオプションです。デフォルトのファイル名は、*bcp.fmt* です。bcp プログラムはデータのコピー時にフォーマット・ファイルを参照できるため、ユーザは以前に指定したフォーマットを対話的に繰り返し指定する必要はありません。このパラメータを使用するのは、以前に作成したフォーマット・ファイルを、今回のコピー・インまたはコピー・アウトにも使用する必要がある場合だけです。このパラメータを指定しない場合、bcp はフォーマット情報を対話的に問い合わせてきます。

-F *firstrow*

入力ファイルからのコピーを開始するローのロー番号です (デフォルトでは先頭のロー)。複数のファイルを使用している場合、このオプションは各ファイルに適用されます。

負荷の高いマルチプロセスのコピーを実行する場合は、このパラメータを使用しないでください。このパラメータを使用すると、通常、bcp は動作に必要な処理が増加し、処理速度が低下します。-F は、単一プロセスの特定のコピーに使用してください。

注意 -F を --skiprows とともに使用することはできません。

-g *id_start_value*

データをコピー・インするときの開始ポイントとして使用する、IDENTITY カラムの値を指定します。

-g フラグと -E フラグを同時に使用することはできません。

-i *input_file*

入力ファイルの名前を指定します。デフォルトは標準入力 (stdin) です。

-I *interfaces_file*

Adaptive Server に接続するときに検索する interfaces ファイルの名前とロケーションを指定します。-I を指定しない場合、bcp は *%SYBASE%¥ini* にある interfaces ファイル (*sql.ini*) を探します。

-J *client_character_set*

クライアントで使用する文字セットを指定します。bcp は、フィルタを使用して *client_charset* と Adaptive Server の文字セット間で入力を変換します。

-J *client_character_set* は、クライアントで使用する文字セットである *client_character_set* とサーバの文字セット間の変換を Adaptive Server に要求します。

-J に引数を指定しないと、文字セット変換が無効になります。この場合、変換は行われません。クライアントとサーバが同じ文字セットを使用する場合に、このパラメータを使用してください。

-J を省略すると、文字セットはプラットフォームのデフォルトに設定されます。デフォルトの文字セットは、クライアントが使用している文字セットと同じであるとはかぎりません。文字セットおよび関連するフラグの詳細については、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。

-K *keytab_file*

DCE での認証に使用する keytab ファイルへのパスを指定します。

-L *lastrow*

入力ファイルからのコピーを終了するローのロー番号です (デフォルトでは最後のロー)。複数のファイルを使用している場合、このオプションは各ファイルに適用されます。

-m *maxerrors*

bcp がコピーをアボートするまでに許容されるエラーの最大数です。bcp は、(データ変換エラーや、null 値を受け付けないカラムに null 値を挿入しようとしたことが原因で) 挿入できないローを破棄し、拒否した各ローを 1 つのエラーと見なします。このオプションを指定しない場合、bcp はデフォルト値 10 を使用します。

複数のパーティションを使用している場合は、maxerrors の値がすべてのファイルに使用されます。

-n

ネイティブの (オペレーティング・システムの) フォーマットを使用して、コピー・オペレーションを実行します。-n パラメータを指定すると、bcp は各フィールドのプロンプトを表示しません。ネイティブ・データ・フォーマットのファイルは、人間には判読できません。

警告！ データ・リカバリやサルベージ、または緊急の問題解決のために、ネイティブ・フォーマットを使用して bcp を実行しないでください。異なるハードウェア・プラットフォーム間、異なるオペレーティング・システム間、または異なるメジャー・リリースの Adaptive Server 間では、ネイティブ・フォーマットの bcp を使用してデータを転送しないでください。フィールド・ターミネータ (-t) やロー・ターミネータ (-r) は、ネイティブ・フォーマットの bcp とともに使用しないでください。予期しない結果となったり、データが破損する可能性があります。ネイティブ・フォーマットの bcp を使用すると、Adaptive Server に再ロードできないフラット・ファイルが作成され、データをリカバリできなくなることがあります。bcp を文字フォーマットで再実行できない場合 (テーブルがトランケートされたり削除されたりした場合、ハードウェアが損傷した場合、データベース・テーブルが削除された場合など)、データをリカバリできません。

-N

IDENTITY カラムをスキップします。データをコピー・インするときに、ホスト・データ・ファイルに IDENTITY カラム値用のプレースホルダが含まれていない場合、またはデータをコピー・アウトするときに、IDENTITY カラムの情報をホスト・ファイルに含めたくない場合に、このパラメータを使用します。

データをコピー・インするときに、-N パラメータと -E パラメータの両方を使用することはできません。

-o *output_file*

出力ファイルの名前を指定します。デフォルトは標準出力 (stdout) です。

-P *password*

Adaptive Server のパスワードを指定します。-P *password* を指定しない場合、bcp はパスワードの入力を求めるプロンプトを表示します。パスワードが NULL の場合は、-P フラグを省略できます。

-Q

コピー・オペレーションで null 入力可能なカラムが含まれている場合に、bcp との下位互換性を実現します。

-r *row_terminator*

ロー・ターミネータを指定します。

警告！ ネイティブ・フォーマットの bcp で、-t パラメータまたは -r パラメータを使用しないでください。予期しない結果となったり、データが破損したりする可能性があります。

コマンド・ラインから -t パラメータまたは -r パラメータでターミネータを指定する場合は、コマンド・プロンプト・シェルに対して特殊な意味を持つ文字をエスケープしてください。特殊文字の前に円記号を付けるか、特殊文字を引用符で囲んでください。詳細については、bcp の例を参照してください。対話モードで bcp を実行している場合は、特殊文字をエスケープする必要はありません。

-R *remote_server_principal*

セキュリティ・メカニズムに定義されたサーバのプリンシパル名を指定します。デフォルトでは、サーバのプリンシパル名はサーバのネットワーク名 (-S パラメータまたは DSQUERY 環境変数で指定) と一致します。サーバのプリンシパル名とネットワーク名が異なる場合は、-R パラメータを使用してください。

-S *server*

接続先の Adaptive Server の名前を指定します。引数なしで -S を指定した場合、bcp は DSQUERY 環境変数で指定されたサーバを使用します。

-t *field_terminator*

デフォルトのフィールド・ターミネータを指定します。

-T *text_or_image_size*

Adaptive Server が送信する text または image データの最大長をバイト単位で指定します。デフォルトは 32K です。text フィールドまたは image フィールドが -T の値またはデフォルト値より大きい場合、bcp はオーバフロー部分を送信しません。

-U *username*

Adaptive Server のログイン名を指定します。*username* を指定していない場合、bcp は現在のユーザのオペレーティング・システムのログイン名を使用します。

-v

bcp の現在のバージョンと版権メッセージを表示して、オペレーティング・システムに戻ります。

-V *security_options*

ネットワーク・ベースのユーザ認証を指定します。このパラメータを指定した場合、ユーザはユーティリティを実行する前にネットワークのセキュリティ・システムにログインする必要があります。この場合、ユーザは -U パラメータでネットワーク・ユーザ名を指定します。-P パラメータで指定されたパスワードは無視されます。

-V の後に *security_options* 文字列を指定することによって、追加のセキュリティ・サービスを有効にできます。指定できる文字は次のとおりです。

- c − データ機密性サービスを有効にする。
- d − クレデンシャル委任を有効にし、クライアント・クレデンシャルをゲートウェイ・アプリケーションに転送する。
- i − データ整合性サービスを有効にする。
- m − 接続を確立するための相互認証を有効にする。
- o − データ・オリジン・スタンピング・サービスを有効にする。
- q − 順序不整合の検出を有効にする。
- r − データ・リプレイの検出を有効にする。

-W

bcp が接続しようとしているサーバが通常のパスワード暗号化と拡張パスワード暗号化のどちらもサポートしていない場合、プレーン・テキスト形式のパスワードを使用した接続再試行を無効にすることを指定します。このオプションを使用すると、CS_SEC_NON_ENCRYPTION_RETRY 接続プロパティが CS_FALSE に設定され、接続の再試行時にプレーン・テキスト形式の (暗号化されていない) パスワードは使用されなくなります。

-X

サーバへの現在の接続で、アプリケーションがクライアント側のパスワード暗号化を使用してログインを開始することを指定します。bcp (クライアント) は、パスワードの暗号化が必要であることをサーバに通知します。サーバは、bcp がパスワードの暗号化に使用する暗号化キーを返送し、パスワードを受け取ると、そのキーを使用してパスワードを認証します。

このオプションでは、サーバでの接続プロパティの設定に応じて、通常のパスワード暗号化が使用される場合もあれば、拡張パスワード暗号化が使用される場合もあります。CS_SEC_ENCRYPTION が CS_TRUE に設定されている場合は、通常のパスワード暗号化が使用されます。CS_SEC_EXTENDED_ENCRYPTION が CS_TRUE に設定されている場合は、拡張パスワード暗号化が使用されます。CS_SEC_ENCRYPTION と CS_SEC_EXTENDED_ENCRYPTION のどちらも CS_TRUE に設定されている場合は、拡張パスワード暗号化が優先的に使用されます。

bcp が失敗すると、パスワードを含むコア・ファイルが作成されます。暗号化オプションを使用していない場合、パスワードは、コア・ファイルにプレーン・テキストで表示されます。暗号化オプションを使用した場合、パスワードは表示されません。

-y *alternate_home_directory*

代替の Sybase ホーム・ディレクトリを設定します。

-Y

bcp in の使用時に、サーバでの文字セット変換を無効にし、代わりにクライアント側で bcp が文字セット変換を実行することを指定します。

注意 bcp out の使用時には、すべての文字セット変換がサーバで実行されます。

-z *language*

サーバが bcp のプロンプトとメッセージの表示に使用する代替言語の公式名です。-z フラグを指定しない場合、bcp はサーバのデフォルト言語を使用します。

言語はインストール時に Adaptive Server に追加できます。インストール後でも、langinst ユーティリティまたは sp_addlanguage ストアド・プロシージャを使用して言語を追加できます。

-z パラメータに不正な言語または認識できない言語を指定すると、次のエラー・メッセージが表示されます。

```
Unrecognized localization object. Using default value 'us_english'.
Starting copy ...
=> warning.
```

-Z *security_mechanism*

接続で使用するセキュリティ・メカニズムの名前を指定します。

セキュリティ・メカニズムの名前は、*%SYBASE%¥%SYBASE_OCS%¥ini* にある *libtcl.cfg* 設定ファイルに定義されています。*security_mechanism* の名前が指定されていない場合は、デフォルトのメカニズムが使用されます。

注意 CS_LIBTCL_CFG プロパティは、代替の *libtcl.cfg* ファイルの名前とパスを指定します。このプロパティの詳細については、『Open Client/Open Server Client Libraries リファレンス・マニュアル』を参照してください。

セキュリティ・メカニズム名の詳細については、『Open Client/Server 設定ガイド Windows 版』の *libtcl.cfg* ファイルの説明を参照してください。

--colpasswd *column_name password*

"set encryption passwd *password* for column *column_name*" を Adaptive Server に送信して、暗号化カラムにパスワードを設定します。これで、他の暗号化カラムが同じキーで暗号化されている場合でも、2 番目のカラムにはパスワードが自動的に適用されません。2 番目のカラムにアクセスするには、パスワードをもう一度指定します。

--hide-vcc

仮想計算カラム (VCC) をデータ・ファイルにコピーしたり、データ・ファイルからコピーしたりしないよう bcp に指示します。bcp out でこのパラメータを使用すると、データ・ファイルには VCC のデータは含まれません。また、bcp in で使用すると、データ・ファイルに VCC のデータを含めることができなくなります。

このオプションを使用した場合、Adaptive Server は仮想計算カラムのデータを計算したり、送信したりしません。

--initstring *"TSQL_command"*

Transact-SQL コマンドを Adaptive Server に送信してから、データが転送されます。

初期化文字列によって発行された結果セットは、エラーが発生しないかぎり暗黙的に無視されます。Adaptive Server からエラーが返された場合、データが転送される前に bcp が停止し、エラー・メッセージが表示されます。

--keypasswd *key_name password*

"set encryption passwd *password* for key *key_name*" を Adaptive Server に送信して、キーを使用してアクセスするすべてのカラムにパスワードを設定します。

--maxconn *maximum_connections*

各バルク・コピー・オペレーションで許可する並列接続の最大数を指定します。たとえば、各オペレーションで許可する並列接続の最大数を 2 に設定するには、次のように入力します。

```
bcp --maxconn 2
```

このパラメータを指定しない場合、bcp はデフォルト値 10 を使用します。

--show-fi

bcp IN または bcp out の使用時に、機能インデックスをコピーするよう bcp に指示します。このパラメータを指定しない場合、Adaptive Server は機能インデックスの値を生成します。

--skiprows *nSkipRows*

指定されたロー数をスキップしてから、入力ファイルからのコピーを開始するよう bcp に指示します。--skiprows の有効範囲は、0 から入力ファイルの実際のロー数までです。無効な値を指定すると、エラー・メッセージが表示されます。

注意 --skiprows を -F オプションとともに使用することはできません。--skiprows を -F オプションとともに使用すると、エラー・メッセージが表示されます。

例

例 1 -c パラメータは、文字フォーマット (すべてのフィールドに char を使用) で publishers テーブルからデータをコピー・アウトします。-t field_terminator パラメータは各フィールドをカンマで終了し、-r row_terminator パラメータは各行を [Return] キーで終了します。bcp は、パスワードの入力を求めるプロンプトだけを表示します。

```
bcp pubs2..publishers out pub_out -c -t , -r ¥r
```

例 2 -C パラメータは、(暗号化カラムがある) publishers テーブルのデータをプレーン・テキストではなく暗号テキストでコピー・アウトします。[Return] キーを押すと、プロンプト画面に表示されたデフォルトが使用されます。publishers テーブルにデータをコピーするときも、同じプロンプトが表示されます。

```
bcp pubs2..publishers out pub_out -C
Password:
Enter the file storage type of field col1 [int]:
Enter prefix length of field col1 [0]:
Enter field terminator [none]:
Enter the file storage type of field col2 [char]:
Enter prefix length of field col2 [0]:
Enter length of field col2 [10]:
Enter field terminator [none]:
Enter the file storage type of field col3 [char]:
Enter prefix length of field col3 [1]:
Enter field terminator [none]:
```

例 3 後で Adaptive Server に再ロードするために、publishers テーブルから *pub_out* というファイルにデータをコピーします。[Return] キーを押すと、プロンプトで指定されたデフォルトが使用されます。publishers テーブルにデータをコピーするときも、同じプロンプトが表示されます。

```
bcp pubs2..publishers out pub_out
Password:
Enter the file storage type of field pub_id [char]:
Enter prefix length of field pub_id [0]:
Enter length of field pub_id [4]:
Enter field terminator [none]:
Enter the file storage type of field pub_name [char]:
Enter prefix length of field pub_name [1]:
Enter length of field pub_name [40]:
Enter field terminator [none]:
Enter the file storage type of field city [char]:
Enter prefix length of field city [1]:
Enter length of field city [20]:
Enter field terminator [none]:
Enter the file storage type of field state [char]:
Enter prefix length of field state [1]:
Enter length of field state [2]:
Enter field terminator [none]:
```

例 4 t1 テーブルの p1 パーティションのデータを、現在のディレクトリの *mypart.dat* ファイルにコピー・アウトします。

```
bcp t1 partition p1 out mypart.dat
```

例 5 保存された *pub_form* フォーマット・ファイルを使用して、Adaptive Server にデータをコピーして戻します。

```
bcp pubs2..publishers in pub_out -f pub_form
```

例 6 この例では、VT200 端末で使用している文字セットで作成したデータ・ファイルを pubs2..publishers テーブルにコピーします。-z フラグは、bcp メッセージをフランス語で表示します。

```
bcp pubs2..publishers in vt200_data -J iso_1 -z french
```

例 7 この例では、Adaptive Server が 4096 バイトのパケット・サイズを使用して 40K の text または image を送信することを指定します。

```
bcp pubs2..publishers out -T 40960 -A 4096
```

例 8 現在のディレクトリの *mypart.dat* ファイルを p1 パーティションの t1 テーブルにコピーします。

```
bcp t1 partition p1 in mypart.dat
```

例 9 パーティション p1、p2、p3 を *¥work2¥data* ディレクトリにあるファイル *a*、*b*、*c* にそれぞれコピーします。

```
bcp t1 partition p1, p2, p3 out ¥work2¥data¥a,
¥work2¥data¥b, ¥work2¥data¥c
```

例 10 ファイル data.first、data.last、data.other をパーティション *p1*、*p2*、*p3* にそれぞれコピーします。

```
bcp t1 partition p1, p2, p3 in data.first, data.last,
data.other
```

例 11 *titles.txt* データが pubs2 titles テーブルに転送されたときに、複写を無効にします。

```
bcp pubs2..titles in titles.txt -- initstring "set replication
off"
```

注意 この例の set replication off コマンドは、Adaptive Server の現在のセッションに限定されるため、bcp の終了後に設定オプションを明示的に再設定する必要はありません。

例 12 暗号化カラム col1 のパスワードを pwd1 に設定します。

```
bcp mydb..mytable out myfile -U uuu -P ppp -colpasswd
db..tbl.col1 pwd1
```

例 13 暗号化カラム col1 のパスワードの入力を求めるプロンプトを設定します。

```
bcp mydb..mytable out myfile -U uuu -P ppp -colpasswd
db..tbl.col1
Enter column db..tbl.col1's password: ***?
```

例 14 “passwordfile” という外部 OS ファイルから、暗号化カラム col1 のパスワードを読み込みます。

```
bcp mydb..mytable out myfile -U uuu -P ppp -colpasswd
db..tbl.col1 < passwordfile
```

例 15 暗号化キー key1 にパスワード pwd1 を設定します。

```
bcp mydb..mytable in myfile -U uuu -p ppp -keypasswd db..key1
pwd1
```

例 16 破棄ファイル *reject_titlesfile.txt* を作成します。

```
bcp pubs2..titles in titlesfile.txt -d reject_
```

例 17 MIT Kerberos のクレデンシャル委任を要求し、クライアント・クレデンシャルを MY_GATEWAY に転送します。

```
bcp -Vd -SMY_GATEWAY
```

例 18 bcp は入力ファイル *titles.txt* の最初の 2 つのローを無視し、3 番目のローからコピーを開始します。

```
bcp pubs2..titles in titles.txt -U username -P password
--skiprows 2
```

例 19 代替の Sybase ホーム・ディレクトリを *C:¥work¥NewSybase* に設定します。

```
bcp tempdb..T1 out T1.out -yC:¥work¥NewSybase -Uuser1
-Psecret -SMYSERVER
```

使用法

- ファイルのコピー・インまたはコピー・アウトに名前付きパイプを使用することはできません。
- --hide-vcc を使用すると、Adaptive Server が仮想計算カラムのデータを転送したり、計算したりしないため、パフォーマンスが向上します。
- bcp の初期化文字列として --initstring を指定して Transact-SQL コマンドを使用できますが、サーバ設定に加えられている可能性のある永続的変更を bcp の実行後に再設定する必要があります。たとえば、別の isql セッションで変更を再設定できます。
- *slice_number* は、Adaptive Server 12.5.x 以前のバージョンとの下位互換性を保つために含まれており、ラウンドロビン方式で分割されたテーブルでのみ使用できます。
- *slice_number* か *partition_name* のいずれかを指定できます。両方を指定することはできません。
- 複数のパーティションとデータ・ファイルを指定できます。各パーティション名またはデータ・ファイルをカンマで区切ります。
- partition_name を指定しない場合、bcp はテーブル全体にコピーします。
- bcp は、データベース・テーブルまたはビューとオペレーティング・システム・ファイル間でデータを高速転送できる便利な方法です。bcp は、さまざまなフォーマットでファイルの読み込みと書き込みを行うことができます。ファイルからコピー・インするときには、bcp はデータを既存のデータベース・テーブルに挿入します。ファイルにコピー・アウトするときには、bcp はファイルの以前の内容を上書きします。
- 処理を完了すると、bcp は正常にコピーされたデータのロー数、コピーに要した合計時間、1 つのローをコピーするのに要した平均時間 (ミリ秒単位)、1 秒当たりにコピーされたロー数を表示します。
- bcp は、対応するターゲット・テーブルのカラムの文字長を超えるエントリを含むローは挿入しません。たとえば、bcp は、300 バイトのフィールドを含むローを、文字カラムの長さが 256 バイトのテーブルには挿入しません。この場合、bcp は変換エラーを表示し、そのローをスキップします。また、トランケートされたデータはテーブルに挿入しません。次のような変換エラーが表示されます。

  ```
  cs_convert: cslib user api layer: common library
  error: The result is truncated because the
  conversion/operation resulted in overflow
  ```

  文字長の要件に違反したデータを記録するには、-e log-file name パラメータを指定して bcp を実行します。bcp は、拒否されたデータのロー番号とカラム番号、エラー・メッセージ、データを、指定したログ・ファイルに記録します。

- 以前のバージョンの bcp を使用するには、*ocs.cfg* ファイルの [bcp] セクションで、CS_BEHAVIOR プロパティを設定します。

  ```
  [bcp]

  CS_BEHAVIOR = CS_BEHAVIOR_100
  ```

  CS_BEHAVIOR を CS_BEHAVIOR_100 に設定していない場合は、bcp 11.1 以降の機能を使用できます。

- bcp が呼び出されたときに、-c、-f、または -n パラメータに値が指定されていない場合は、bcp プロンプトがファイル記憶タイプを要求します。ファイル記憶タイプは Adaptive Server で有効な任意のデータ型です。bigdatetime および bigtime Adaptive Server の記憶タイプは次のように指定されます。

記憶タイプ	テーブルでのデータ型
A	bigdatetime
B	bigtime

- bigdatetime データ型または bigtime データ型を使用して、bcp フォーマット・ファイルに次のデータ型を指定できます。

表 *A-1:* ホスト・ファイルのデータ型の記憶フォーマット

記憶フォーマット	Adaptive Server データ型
SYBBIGDATETIME	bigdatetime
SYBBIGTIME	bigtime

-d オプションの使用

- -d オプションの指定が適用されるのは、バルク・コピー・インの場合だけです。バルク・コピー・アウトで使用した場合は、暗黙的に無視されます。
- 複数の入力ファイルを使用する場合、エラーのあるローを含む入力ファイルごとに、破棄ファイルが 1 つずつ作成されます。
- エラーの最大許容数に達すると、bcp は失敗したローのログが取られるまで、バッチの最初からすべてのローのオペレーションを停止します。
- -b オプションを使用すると、バッチ・サイズが自動的に調整されます。次の場合に警告メッセージが表示されます。
  - -b *batchsize* は指定されているが、バッチまたはローのサイズが大きすぎるため、バッチのすべてのローをメモリに保持できない場合
  - -b *batchsize* を指定していない場合

インデックスまたはトリガのあるテーブルのコピー

- bcp は、インデックスまたはトリガが関連付けられていないテーブルにデータをロードするために最適化されています。bcp は、インデックスやトリガを使用せずに、ロギングを最小限にすることで、データをテーブルに最大限の速度でロードします。ページの割り付けはログを取られますが、ローの挿入はログを取られません。

  1 つ以上のインデックスまたはトリガを持つテーブルにデータをコピーするときには、bcp の低速バージョンが自動的に使用され、ローの挿入のログが取られます。これには、create table 文の一意整合性制約を使用して暗黙的に作成されたインデックスも含まれます。ただし、bcp は、テーブルに定義されている他の整合性制約は適用しません。

  高速バージョンの bcp はログを取らずにデータを挿入するため、システム管理者またはデータベース所有者は、sp_dboption DBNAME, “select into/bulkcopy”, true を最初に設定しておく必要があります。オプションが true でないときに、インデックスやトリガのないテーブルにデータをコピーしようとすると、Adaptive Server はエラー・メッセージを生成します。データをファイルにコピー・アウトする場合や、インデックスまたはトリガを含むテーブルにデータをコピー・インする場合は、このオプションを設定する必要はありません。

  注意　bcp は、インデックスまたはトリガを持つテーブルへの挿入をログに取るため、ログが非常に大きくなる可能性があります。バルク・コピーの完了後、dump database を使用してデータベースをバックアップしてから、dump transaction を使用してログをトランケートしてください。

- select into/bulkcopy オプションがオンになっているときは、トランザクション・ログをダンプすることはできません。dump transaction を発行するとエラー・メッセージが表示され、代わりに dump database を使用するよう指示されます。

  警告！　select into/bulkcopy フラグをオフにする前に、必ずデータベースをダンプしてください。ログが取られていないデータをデータベースに挿入し、dump database を実行する前に dump transaction を実行した場合は、そのデータをリカバリすることはできません。

- dump database が実行されている間、ログが取られていない bcp は実行速度が低下します。
- 表 A-2 では、コピー・インのときに bcp がどのバージョンを使用するのかを示し、select into/bulkcopy オプションに必要な設定を示しています。また、トランザクション・ログが保持されるかどうか、またダンプできるかどうかも示しています。

表 *A-2:* 高速 *bcp* と低速 *bcp* の比較

	select into/ bulkcopy が on の場合	**select into/ bulkcopy が off の場合**
高速 bcp (ターゲット・テーブルにインデックスまたはトリガがない場合)	実行可能 dump transaction 実行不可	bcp dump transaction 実行不可
低速 bcp (1 つまたは複数のインデックスまたはトリガがある場合)	実行可能 dump transaction 実行不可	実行可能 dump transaction 実行可能

- デフォルトでは、新しく作成されたデータベースの select into/bulkcopy オプションはオフです。デフォルトを変更するには、model データベースでこのオプションをオンにします。

注意 インデックスまたはトリガを持つテーブルにデータをコピーする場合、パフォーマンスが大幅に低下する可能性があります。多数のローをコピー・インする場合は、すべてのインデックスとトリガを削除し、データベース・オプションの設定、テーブルへのデータのコピー、インデックスとトリガの再作成を行ってからデータベースをダンプすると、処理速度が上がることがあります。ただし、インデックスとトリガを構成するために、データに必要な格納領域の約 2.2 倍の追加のディスク領域を割り付ける必要があります。

---

bcp プロンプトに対する応答

-n (ネイティブ・フォーマット) パラメータまたは -c (文字フォーマット) パラメータを使用して、データをコピー・インまたはコピー・アウトする場合、-P パラメータでパスワードを指定していないと、パスワードの入力だけを求めるプロンプトが表示されます。-n、-c、または -f *formatfile* パラメータのいずれも指定していない場合は、テーブルの各フィールドに関する情報の入力を求めるプロンプトが表示されます。

- 各プロンプトでは、デフォルト値は角カッコで表示されます。[Return] キーを押すと、この値を選択できます。プロンプトには、次のものがあります。
  - ファイル記憶タイプ。character データ型または Adaptive Server で有効な任意のデータ型。
  - プレフィクス長 (後続のデータの長さをバイト単位で示す整数)。
  - ファイル内の NULL でないフィールドのデータの記憶長。
  - フィールド・ターミネータ (任意の文字列)。
  - numeric データ型と decimal データ型の位取りと精度。

  ロー・ターミネータは、テーブルまたはファイルの最後のフィールドのフィールド・ターミネータです。

- 角カッコ内のデフォルト値は、該当するフィールドのデータ型として適切な値を表しています。ファイルへコピー・アウトする場合の空き領域の最適な使用方法は、次のとおりです。
  - デフォルトのプロンプトを使用する。
  - すべてのデータをそのテーブルのデータ型でコピーする。
  - 指定どおりにプレフィクスを使用する。
  - ターミネータを使用しない。
  - デフォルトの長さを使用する。

  表 A-3 に、デフォルトおよび代替可能な応答を示します。

表 *A-3: bcp* プロンプトのデフォルトと応答

プロンプト	デフォルト設定	可能な応答
ファイル記憶タイプ	次のフィールドを除くほとんどのフィールドに対してデータベースの記憶タイプを使用する。 *varchar* には *char* *varbinary* には *binary*	人間が判読できるファイルの作成または読み込みを行う場合は char。暗黙の変換がサポートされている場合は Adaptive Server の任意のデータ型。
プレフィクス長	• 0 － (記憶タイプではなく) データ型 (*char* データ型とすべての固定長データ型) で定義されたフィールドの場合 • 1 － その他のほとんどのデータ型の場合 • 2 － char として保存される binary と varbinary の場合 • 4 － text と image の場合	0 － プレフィクスが不要な場合。他のすべての場合ではデフォルトの使用を推奨。
記憶長	char と varchar の場合は、定義されている長さを使用する。char として保存される binary と varbinary の場合は、デフォルトを使用する。他のすべてのデータ型では、トランケーションやデータのオーバフローを避けるために必要な最大長を使用する。	デフォルト値またはそれ以上の値を推奨。

プロンプト	デフォルト設定	可能な応答
フィールド・ターミネータまたはロー・ターミネータ	なし	30 文字以内、または次のいずれか。 • ¥t タブ • ¥n 改行 • ¥r 復帰改行 • ¥0 null ターミネータ • ¥ 円記号

- bcp は、ネイティブ (データベース) データ型、または暗黙の変換がサポートされている任意のデータ型として、データをファイルにコピー・アウトできます。bcp は、ユーザ定義のデータ型をその基本データ型または暗黙の変換がサポートされている任意のデータ型としてコピーします。詳細については、『Open Client DB-Library/C リファレンス・マニュアル』の「dbconvert」を参照してください。

  注意 すべてのバージョンで同じデータ型をサポートしているわけではないため、異なるバージョンの Adaptive Server からデータをコピーするときは注意してください。

- プレフィクス長は、各データ値の長さをバイト単位で表現する 1 バイト、2 バイト、または 4 バイトの整数です。プレフィクス長は、ホスト・ファイルのデータ値の直前に指定します。
- データベース内で char、nchar、binary として定義されるフィールドは、データベース内で定義された全長に達するまで、常にスペース (binary の場合は null バイト) が埋め込まれます。*timestamp* データは、binary(8) として扱われます。

  varchar フィールドと varbinary フィールドのデータがコピー・アウト用に指定した長さより長い場合、bcp はファイルのデータを指定された長さに暗黙的にトランケートします。
- フィールド・ターミネータ文字列は、30 文字まで指定できます。最も一般的なターミネータは、タブ (「¥t」と入力し、最後のカラム以外のすべてのカラムに使用する) と改行 (「¥n」と入力し、ローの最後のフィールドに使用する) です。その他、“¥0” (null ターミネータ)、“¥” (円記号)、“¥r” (復帰改行) があります。ターミネータを選択するときは、使用している文字データで同じパターンが出現しないことを確認してください。たとえば、タブを含む文字列でタブ・ターミネータを使用すると、bcp は文字列の最後を表すタブを識別できません。bcp はターミネータとして指定された文字列が最初に出現したときに、常にそれをターミネータと判断するため、この例ではターミネータではないものをターミネータと判断することになります。

ターミネータまたはプレフィクスが存在する場合は、転送されるデータの実際の長さに影響します。ファイルにコピー・アウトするエントリの長さが記憶長より短い場合は、その直後にターミネータまたは次のフィールドのプレフィクスが続きます。この場合、エントリに記憶長分の埋め込みは行われません (char、nchar、binary データは、Adaptive Server から返されるときに、既にいっぱいの長さまで埋め込みが行われています)。

ファイルからコピー・インするときは、「長さ」プロンプトで指定されたバイト数がコピーされるか、ターミネータが検出されるまでデータが転送されます。指定された長さのバイト数の転送が終了すると、残りのデータはターミネータが検出されるまでフラッシュされます。ターミネータがない場合、テーブルの記憶領域の長さが使用されます。

- 表 A-4 と 表 A-5 に、ファイル内の情報のプレフィクス長、ターミネータ、フィールド長の関係を示します。“P” は格納されているテーブルのプレフィクス、“T” はターミネータ、ダッシュ (“--”) は追加された領域をそれぞれ表します。“...” は、各フィールドに対してパターンを繰り返すことを示します。各カラムのフィールド長は 8 バイトです。“string” は、それぞれ 6 文字のフィールドを表します。

表 *A-4: Adaptive Server の char データ*

	プレフィクス長 0	プレフィクス長 1、2、または 4
ターミネータなし	string--string--	Pstring--Pstring--
ターミネータ	string--Tstring--T	Pstring--TPstring--T

表 *A-5: char* 記憶領域に変換された他のデータ型

	プレフィクス長 0	プレフィクス長 1、2、または 4
ターミネータなし	string--string--	PstringPstring
ターミネータあり	stringTstringT	PstringTPstringT

- カラムのファイル記憶タイプおよび長さは、データベース・テーブルのカラムのタイプおよび長さと同じである必要はありません。ただし、コピー・インされたタイプとフォーマットがデータベース・テーブルの構造と矛盾する場合、コピーは失敗します。
- 通常、ファイル記憶長は、カラムに転送されるデータの最大サイズ (ターミネータとプレフィクスを除く) を示します。
- テーブルにデータをコピーする場合は、bcp はカラムに対して定義されているデフォルトとユーザ定義のデータ型を調べます。ただし、bcp は最大限の速度でデータをロードするためにルールを無視します。
- bcp は、null 値を含むことができるデータ・カラムを可変長と見なすため、プレフィクス長またはターミネータのいずれかを使用して、各データ・ローの長さを示してください。

- ネイティブ・フォーマットでホスト・ファイルに書き込まれたデータは、その精度をすべて保持します。datetime 値と float 値は、文字フォーマットに変換されるときにも精度をすべて保持します。Adaptive Server は、money 値を通貨単位の 1 万分の 1 の精度まで保持します。ただし、money 値が文字フォーマットに変換されるときには、文字フォーマット値は近似値 2 桁しか記録されません。
- 文字フォーマットのデータをファイルからデータベース・テーブルにコピーする前に、『ASE リファレンス・マニュアル』のデータ型に関する項で説明されているデータ型の入力規則を確認してください。bcp を使用してデータベースにコピーする文字データは、この規則に従っていなければなりません。区切り文字のない *(yy)yymmdd* 形式の日付は、年が最初に指定されていないと、オーバフロー・エラーになることがあります。
- 使用している端末とは異なる端末を使用するサイトにホスト・データ・ファイルを送信する場合は、ファイルを作成するときに使用した *datafile_charset* を通知してください。

メッセージ

- `Error in attempting to load a view of translation tables.`

  -q パラメータで指定した文字変換ファイルが見つからないか、入力したファイル名に誤りがあります。
- `Unable to open the discard-file` ***`discardfilename`***`.`
- `I/O error while writing the bcp` ***`discardfilename`***`.`
- `Unable to close the file` ***`discardfilename`***`. Data may not have been copied.`

パーミッション

bcp を使用するには、Adaptive Server アカウント、データベース・テーブルに対する適切なパーミッション、および転送で使用するオペレーティング・システム・ファイルが必要です。

- データをテーブルにコピーするには、そのテーブルに対する insert パーミッションが必要です。
- テーブルをオペレーティング・システム・ファイルにコピーするには、次のテーブルに対する select パーミッションが必要です。
  - コピーするテーブル
  - sysobjects
  - syscolumns
  - sysindexes

cpre

説明

cpre は、C ソース・プログラムをプリコンパイルして、ターゲット・ファイル、リスティング・ファイル、isql ファイルを生成します。

構文

```
cpre
    [-C compiler]
    [-D database_name]
    [-F fips_level]
    [-G [isql_file_name]]
    [-H]
    [-I include_path_name]...
    [-J charset_locale_name]
    [-K syntax_level]
    [-L [listing_file_name]]
    [-N interface_file_name]
    [-O target_file_name]
    [-P password]
    [-S server_name]
    [-T tag_id]
    [-U user_id]
    [-V version_number]
    [-Z language_locale_name]
    [@ options_file]
    [-a] [-b] [-c] [-d] [-e] [-f] [-h] [-l] [-m] [-p] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
    filename[.ext]
```

注意　スラッシュ (/) またはハイフン (-) のどちらを使用してもオプションをフラグできます。したがって、cpre -I と cpre /I は同じことを表します。

パラメータ

-C *compiler*

次のように、対象のホスト言語コンパイラの値を指定します。

- “ANSI_C” − ANSI C コンパイラ。
- “MSVC” − Microsoft Visual C コンパイラ。“MSVC” プリコンパイラの出力では、1K バイトより長い文字列は生成されません。

-D *database_name*

解析対象のデータベースの名前を指定します。このオプションは、プリコンパイル時に SQL のセマンティックをチェックする場合に使用します。-G を指定すると、use *database* コマンドが *filename.sql* ファイルの先頭に追加されます。このオプションを指定しない場合、プリコンパイラは Adaptive Server のデフォルト・データベースを使用します。

-F *fips_level*

指定の準拠レベルを調べます。プリコンパイラは、SQL89 または SQL92E を調べることができます。

-G *isql_file_name*

該当する SQL 文のストアド・プロシージャを生成し、isql によるデータベースへの入力に使用するファイルに保存します。複数の入力ファイルがある場合は、-G を使用できますが、引数を指定することはできません。

複数の入力ファイルがある場合、または引数を指定しない場合は、デフォルトのターゲット・ファイル名は、*.isql* 拡張子を付加した (または入力ファイル名の拡張子をこの拡張子に置き換えた) 入力ファイル名になります。

ストアド・プロシージャのタグ ID を指定するときは、-Ttag_id オプションも参照してください。

-G オプションを使用しない場合、ストアド・プロシージャは生成されません。

-H

高可用性 (HA: High Availability) フェールオーバ機能を使用してコードを生成します。

-I *include_path_name*

Embedded SQL がインクルード・ファイルを検索するディレクトリを完全なパス名で指定します。このオプションは何回でも指定できます。Embedded SQL は、コマンド・ラインに指定された順序で複数のディレクトリを検索します。このオプションを使用しない場合、デフォルトは Sybase リリース・ディレクトリの *¥include* ディレクトリと現在の作業ディレクトリです。

-J *charset_locale_name*

プリコンパイルするソース・ファイルの文字セットを指定します。このオプションの値は、ロケール・ファイル内のエントリに対応するロケール名でなければなりません。-J を指定しない場合、プリコンパイラはソース・ファイルがプリコンパイラのデフォルトの文字セットを使用しているものと解釈します。

デフォルトとして使用する文字セットを決定するために、プリコンパイラはロケール名を調べます。CS-Library は、次の順序で検索します。

1 LC_ALL

2 LANG

これらのロケール値がいずれも定義されていない場合、CS-Library は「デフォルト」のロケール名を使用します。

プリコンパイラは、*%SYBASE%¥locales* ディレクトリにある *locales.dat* ファイルでロケール名を探し、そのロケール名に関連付けられている文字セットをデフォルトの文字セットとして使用します。

-K *syntax_level*
実行する構文チェックのレベルを指定します。選択肢は次のとおりです。

- none (デフォルト)
- syntax
- semantic

syntax または semantic を使用する場合は、Embedded SQL が Adaptive Server に接続できるように、-U、-P、-S、-D の各オプションも指定する必要があります。

このオプションを使用しない場合、プリコンパイラはサーバに接続しないか、ターゲット・ファイルを生成するために必要な部分を除く入力ファイルの SQL 構文チェックを実行します。

-L *listing_file_name*
1 つまたは複数のリスティング・ファイルを生成します。リスティング・ファイルは、行番号が付けられた各行の後に、エラーがある場合は該当するエラー・メッセージが続く入力ファイルの 1 つのバージョンです。複数の入力ファイルがある場合は -L を使用できますが、引数を指定することはできません。

複数の入力ファイルがある場合、または引数を指定しない場合は、デフォルトのリスティング・ファイル名は、*.lis* 拡張子を付加した (または入力ファイル名の拡張子をこの拡張子に置き換えた) 入力ファイル名になります。

このオプションを指定しない場合は、リスティング・ファイルは生成されません。

-N *interface_file_name*
プリコンパイラに対して interfaces ファイルの名前 (*sql.ini*) を指定します。

-O *target_file_name*
ターゲット・ファイルとなる出力ファイルの名前を指定します。複数の入力ファイルがある場合は、このオプションを使用できません (デフォルトのターゲット・ファイル名が割り当てられます)。このオプションを使用しない場合、デフォルトのターゲット・ファイル名は、*.cbl* 拡張子を付加した (または入力ファイル名の拡張子をこの拡張子に置き換えた) 入力ファイル名になります。

-P *password*
プリコンパイル時に SQL 構文チェックを実行するための Adaptive Server パスワードを指定します。-P を引数なしで使用するか、引数としてキーワード NULL を指定すると、null (“ ”) パスワードが指定されます。-P を使用せずに -Uuser_id オプションを使用した場合、プリコンパイラはパスワードの入力を求めるプロンプトを表示します。このオプションは -G フラグとともに使用してください。

-S *server_name*

プリコンパイル時に SQL 構文チェックを実行する場合の Adaptive Server の名前を指定します。このオプションを使用しない場合は、DSQUERY 環境変数からデフォルトの Adaptive Server 名が取得されます。DSQUERY が設定されていない場合には、SYBASE がサーバの名前として使用されます。

-T *tag_id* (-G とともに使用)

生成されるストアド・プロシージャ・グループ名の最後に付加するタグ ID (3 文字以内) を指定します。

たとえば、-Tdbg をコマンドの一部として入力した場合、生成されるストアド・プロシージャには、タグ ID として *dbg* が付加された入力ファイルの名前 (*program_dbg;1*、*program_dbg;2*, など) が割り当てられます。

プログラマはタグ ID を使用することによって、使用中の可能性がある生成済みの既存のストアド・プロシージャに影響を与えずに、既存のアプリケーションに対する変更をテストできます。

このオプションを使用しない場合は、ストアド・プロシージャ名にタグ ID は追加されません。

-U *user_id*

Adaptive Server のユーザ ID を指定します。このオプションを使用すると、プリコンパイル時に SQL 構文を検査できます。このオプションを使用すると、プリコンパイラは解析だけを目的として SQL 文をサーバに渡します。サーバが構文エラーを検出すると、エラーがレポートされてコードは生成されません。-P[password] を使用していない場合は、パスワードの入力を求めるプロンプトが表示されます。

-K、-P、-S、-D も参照してください。

-V *version_number*

Client-Library のバージョン番号を指定します。COBOL の場合、バージョン番号は *cobpub.cbl* の値のいずれかと一致する必要があります。このオプションを使用しない場合、デフォルトはプリコンパイラで使用できる Client-Library の最新バージョン (Open Client/Open Server バージョン 15.5 の場合は CS_VERSION_155) になります。

-Z *language_locale_name*

プリコンパイラがメッセージに使用する言語と文字セットを指定します。-Z を指定しない場合、プリコンパイラはそのデフォルトの言語と文字セットをメッセージに使用します。

プリコンパイラは、メッセージ用のデフォルトとして使用する言語と文字セットを次のようにして決定します。

1　ロケール名を探します。CS-Library は次の順序で情報を検索します。

   a　LC_ALL

   b　LANG

   これらのロケール値がいずれも定義されていない場合、CS-Library は「デフォルト」のロケール名を使用します。

2　*locales.dat* ファイル内でロケール名を探して、そのロケール名に関連付けられた言語と文字セットを確認します。

3　手順 2 で調べた言語と文字セットに対応する、ローカライズされたメッセージと文字セットの情報をロードします。

@options_file

上記のコマンド・ライン引数のいずれかを含んでいるファイルを指定するために使用します。プリコンパイラは、すでに指定されている引数に加えてこのファイルに含まれている引数を読み込みます。*@options_file* で指定するファイル内にプリコンパイルするファイルの名前が含まれている場合は、この引数はコマンド・ラインの最後に置いてください。

-a

トランザクション間で、カーソルをオープンしたままにできるようにします。このオプションを使用しない場合、カーソルは set close on endtran on が有効な場合と同様に動作します。これは ANSI 準拠の動作です。詳細については、『ASE リファレンス・マニュアル』を参照してください。

-b

fetch 文で一般的に使用されるホスト変数アドレスの再バインドを無効にします。このオプションを使用しない場合は、Embedded SQL/C プログラム内で別の方法で指定しないかぎり、fetch 文が出現するたびに再バインドが行われます。

-b オプションは、Embedded SQL プリコンパイラのバージョンによって次のように異なります。

- 11.1 以降のバージョンの cpre では、-b オプションを使用して宣言がプリコンパイルされているカーソルのすべての fetch 文に norebind 属性が適用されます。
- 10.0 以前のバージョンの cpre では、カーソルが宣言された場所に関係なく、-b オプションを使用してプリコンパイルされた各 Embedded SQL ソース・ファイル内のすべての fetch 文に norebind 属性が適用されます。

-c

ct_debug に対する呼び出しを生成することによって Client-Library のデバッグ機能をオンにします。

このオプションは、アプリケーションの開発時には役立ちますが、アプリケーションを最終的に配布するときにはオフにしてください。このオプションを適切に機能させるには、*%SYBASE%¥%SYBASE_OCS¥devlib* ディレクトリにあるライブラリと、*%SYBASE%¥%SYBASE_OCS¥devdll* ディレクトリにある DLL にアプリケーションをリンクして実行する必要があります。

-d

区切り識別子 (二重引用符で囲まれた識別子) をオフにし、SQL 文内の引用符で囲まれた文字列を文字リテラルとして扱えるようにします。

-e

exec sql connect 文を処理するときに、外部設定ファイルを使用して接続を設定するよう Client-Library に指示します。-x オプションと、『Open Client Client-Library/C リファレンス・マニュアル』の CS_CONFIG_BY_SERVERNAME プロパティも参照してください。

このオプションを使用しない場合、プリコンパイラは Client-Library の関数呼び出しを生成して接続を設定します。外部設定ファイルの詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

-f

ANSI FIPS 準拠の検査を行うための FIPS フラガをオンにします。

-h

スレッドセーフ・コードを生成します。

-l

#line ディレクティブを生成しないようにします。

-m

アプリケーションを Sybase のオートコミット・モードで実行します。このモードではトランザクションが連鎖しません。明示的な begin トランザクションと end トランザクションが必要となります。これらがない場合は、各文が即座にコミットされます。このオプションを指定しない場合、アプリケーションは ANSI 形式の連鎖トランザクション・モードで実行されます。

-p
このオプションを使用すると、入力ホスト変数を持つモジュール内の SQL 文ごとに個別のコマンド・ハンドルが生成され、各コマンド・ハンドルで継続バインドが有効になります。このオプションにより、入力パラメータが指定されたコマンドを繰り返し実行するときのパフォーマンスが向上します。ただし、格納領域の使用量が増加し、各コマンドの初回の実行時間が長くなります。

ホスト文字列変数が空のときに、NULL 文字列の代わりに空の文字列を挿入しないと動作しないアプリケーションは、-p オプションがオンになっていると動作しません。継続バインドを実装しているので、Embedded SQL は Client-Library プロトコル (NULL 文字列を挿入する) を回避することができません。

-r
繰り返し読み出しを無効にします。このオプションを使用していない場合、connect 文の最中に実行される、set transaction isolation level 3 文が生成されます。デフォルトの独立性レベルは 1 です。

-s
静的関数宣言をインクルードします。

-u
ANSI 形式のバインドを無効にします。

-v
(プリコンパイルを実行せずに) プリコンパイラのバージョン情報だけを表示します。

-w
警告メッセージの表示をオフにします。

-x
外部設定ファイルを使用します。『Open Client Client-Library/C リファレンス・マニュアル』に記載されている CS_EXTERNAL_CONFIG プロパティと、『Open Client Embedded SQL/C プログラマーズ・ガイド』に記載されている INITIALIZE_APPLICATION 文の説明を参照してください。

-y
S_TEXT データ型と CS_IMAGE データ型を入力ホスト変数として使用できるようにします。実行時に、データはサーバに送信される文字列に直接挿入されます。サポートされるのは静的 SQL 文だけです。動的 SQL の入力パラメータとして、text と image を使用することはできません。この引数のコマンド文字列への置換は、-y コマンド・ライン・オプションが使用されたときだけ実行されます。

filename[.ext]
ESQL/C ソース・プログラムの入力ファイル名を指定します。ファイル名の形式と長さは、規則に違反しないかぎり、どのようなものでもかまいません。

例

例 1 プリコンパイラを実行します (ANSI 準拠)。

```
cpre program.pc
```

例 2 生成されたストアド・プロシージャと FIPS フラグを使用して、プリコンパイラを実行します (ANSI 準拠)。

```
cpre -G -f program1.pc
```

例 3 トランザクション間でカーソルがオープンしたままの状態で、入力ファイルに対してプリコンパイラを実行します (ANSI 非準拠)。

```
cpre -a program1.pc
```

例 4 プリコンパイラのバージョン情報だけを表示します。

```
cpre -v
```

例 5 最高レベルの SQL チェックを指定して、プリコンパイラを実行します。

```
cpre -K SEMANTIC -Uuser_id -Ppassword -Sserver_name -Dpubs2
example1.pc
```

使用法

- cpre コマンドのデフォルトは、ANSI 標準の動作に対して設定されます。
- -a、-c、-f、-m、-r、-V オプションは connect 文だけに影響します。ソース・ファイルに connect 文が含まれていない場合、または -e か -x を使用する場合は、これらのオプションは影響しません。
- オプションは、引数の前にスペースがあってもなくてもかまいません。たとえば、次のどちらでも問題ありません。
  - -Tdbg
  - -T dbg
- プリコンパイラは複数の入力ファイルを処理できます。-O *target_file_name* オプションを使用しなくてもかまいませんが、その場合はデフォルトのターゲット・ファイル名を使用する必要があります (上記の「ターゲット・ファイル」の説明を参照)。-G[isql_file_name] オプションを使用する場合は、引数を指定できません。デフォルトの isql ファイル名は、*first_input_file.sql*、*second_input_file.sql* などです。-L[listing_file_name] オプションを使用する場合は、引数を指定することはできません。デフォルトのリスティング・ファイル名は、*first_input_file.lis*、*second_input_file.lis* などです。
- デフォルトでは、cpre はインジケータ変数の ANSI 形式のバインド (CS_ANSI_BINDS) を有効にする ct_options に対する呼び出しを生成します。null 入力可能なホスト変数を表すインジケータ変数 (*columns*) が使用できない場合、Client-Library は致命的な実行時エラーを生成し、使用中のアプリケーションをアボートします。これらの問題は、cpre とともに -u を使用することで回避できます。*ocs.cfg* ファイルで CS_ANSI_BINDS を cs_false に設定して、ANSI 形式のバインドを無効にすることもできます。

アプリケーションの開発

この項では、Embedded SQL アプリケーションの開発で最も一般的に使用される手順について説明します。この手順は、稼働条件に合うように適応させることが必要な場合もあります。これらの手順は、DOS コマンド・プロンプトから実行してください。

1　構文チェックとデバッグを行うために、-c、-Ddatabase_name、-P[password]、-Sserver_name、-K[SYNTAX| SEMANTIC]、-Uuser_id の各オプションを使用して、プリコンパイラを実行します。-G[isql_file_name] は使用しないでください。プログラムのコンパイルとリンクを実行して、構文が正しいかどうか確認します。

2　必要な修正をすべて行います。-Ddatabase_name、-G[isql_file_name]、-Ttag_id の各オプションを使用してプリコンパイラを実行し、テスト・プログラム用のタグ ID を持つストアド・プロシージャを生成します。テスト・プログラムをコンパイルしてリンクします。次のコマンドを使用して、ストアド・プロシージャをロードします。

```
isql -Ppassword -Sserver_name -Uuser_id -iisql_file_name
```

プログラム上でテストを実行します。

3　修正版のプログラムに対して、-Ddatabase_name と -G[isql_file_name] の各オプションを使用して (-T オプションは使用しない) プリコンパイラを実行します。プログラムをコンパイルしてリンクします。次のコマンドを使用して、ストアド・プロシージャをロードします。

```
isql -Ppassword -Sserver_name -Uuser_id -iisql_file_name
```

これで、最終的な配布用プログラムを実行する準備が完了しました。

プリコンパイラがサーバの名前を確認する方法

プリコンパイル時に Adaptive Server に接続することによって、プリコンパイル時に追加で構文チェックを実行できます。プリコンパイラは、次の 3 つの方法のいずれかを使用してサーバの名前を調べます。

- cpre コマンド・ラインで -S オプションを使用する
- DSQUERY 変数を設定する
- デフォルト値 “SYBASE” を使用する

-S オプションは、DSQUERY によって設定された値を上書きします。

プリコンパイル・コマンド・ラインでサーバを選択するには、次の構文を使用します。

```
cpre -Usa -P -Sserver_name
```

別の方法として、接続呼び出しまたは接続文のサーバ名を無視することもできます。この場合、*server_name* は DSQUERY 環境変数のランタイム値から値を取得します。アプリケーション・ユーザが DSQUERY を設定していない場合、サーバ名のランタイム値はデフォルトの “SYBASE” になります。DSQUERY の詳細については、『Open Client/Server 設定ガイド Windows 版』を参照してください。

cpre のデフォルト

表 A-6 に、cpre と cobpre のオプションとデフォルトを示します。

表 *A-6: cpre と cobpre のデフォルト*

オプション	オプションを使用しない場合のデフォルト
-C *compiler*	COBOL の場合は mf_byte コンパイラ。C の場合は ANSI-C。
-D *database_name*	Adaptive Server のデフォルト・データベース。
-F *fips_level*	(FIPS フラグは使用不可)
-G [*isql_file_name*]	ストアド・プロシージャは生成されない。
-I *include_path_name*	デフォルト・ディレクトリは Sybase リリース・ディレクトリの *¥include* サブディレクトリ。
-J *charset_locale_name*	[プラットフォームによって異なる]
-K [syntax | semantic | none]	syntax と semantic のどちらも選択しない場合、デフォルト設定は “None”。
-L [*listing_file_name*]	リスティング・ファイルは生成されない。
-N *interface_file_name*	Sybase リリース・ディレクトリの *¥ini* サブディレクトリにある *sql.ini* ファイル。
-O *target_file_name*	デフォルトのターゲット・ファイル名は、拡張子 *.cbl* または *.c* が付加された (または入力ファイル名の拡張子をこれらの拡張子に置き換えた) 入力ファイル名。
-P *password*	-U*user_id* を使用しないかぎり、パスワード入力のプロンプトは表示されない。
-S server_name	デフォルトの Adaptive Server 名は DSQUERY 環境変数から取得される。
-T *tag_id*	-G を使用して生成されるストアド・プロシージャ名にはタグ ID は追加されない。
-U *user_id*	なし。
-V *version_number*	CS_VERSION_125 (バージョン 12.5.x の場合) CS_VERSION_150 (バージョン 15.0 の場合) CS_VERSION_155 (バージョン 15.5 の場合)
-Z *language_locale_name*	[プラットフォームまたは環境によって異なる]

cobpre

説明

cobpre は、COBOL ソース・プログラムをプリコンパイルして、ターゲット・ファイル、リスティング・ファイル、isql ファイルを生成します。

構文

cobpre
[-C *compiler*]
[-D *database_name*]
[-F *fips_level*]
[-G [*isql_file_name*]]
[-I *include_path_name*]
[-J *charset_locale_name*]
[-K *syntax_level*]
[-L [*listing_file_name*]]
[-N *interface_file_name*]
[-O *target_file_name*]
[-P *password*]
[-S *server_name*]
[-T *tag_id*]
[-U *user_id*]
[-V *version_number*]
[-Z *language_locale_name*]
[@ *options_file*]
[-a] [-b] [-c] [-d] [-e] [-f] [-l] [-m] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
filename[.ext]

注意　スラッシュ (/) とハイフン (-) のどちらを使用しても、オプションをフラグできます。したがって、cobpre -I と cobpre /I は同じことを表します。

パラメータ

-C *compiler*
次のように、対象のホスト言語コンパイラの値を指定します。

- “mf_byte” − バイト整列データを使用する Micro Focus COBOL (-C NOIBMCOMP)。
- “mf_word” − ワード整列データを使用する Micro Focus COBOL (-C IBMCOMP)。

-D *database_name*
解析対象のデータベースの名前を指定します。このオプションは、プリコンパイル時に SQL のセマンティックをチェックする場合に使用します。-G を指定すると、use *database* コマンドが *filename.sql* ファイルの先頭に追加されます。このオプションを指定しない場合、プリコンパイラは Adaptive Server のデフォルト・データベースを使用します。

-F *fips_level*
指定の準拠レベルを調べます。プリコンパイラは、SQL89 または SQL92E を調べることができます。

-G *isql_file_name*

該当する SQL 文のストアド・プロシージャを生成し、isql によるデータベースへの入力に使用するファイルに保存します。複数の入力ファイルがある場合は、-G を使用できますが、引数を指定することはできません。

複数の入力ファイルがある場合、または引数を指定しない場合は、デフォルトのターゲット・ファイル名は、*.isql* 拡張子を付加した (または入力ファイル名の拡張子をこの拡張子に置き換えた) 入力ファイル名になります。

ストアド・プロシージャのタグ ID を指定するときは、-Ttag_id オプションも参照してください。

-G オプションを使用しない場合、ストアド・プロシージャは生成されません。

-I *include_path_name*

Embedded SQL がインクルード・ファイルを検索するディレクトリを完全なパス名で指定します。このオプションは何回でも指定できます。Embedded SQL はコマンド・ラインに指定された順に複数のディレクトリを探します。このオプションを使用しない場合、デフォルトは Sybase リリース・ディレクトリの *¥include* ディレクトリと現在の作業ディレクトリです。

-J *charset_locale_name*

プリコンパイルするソース・ファイルの文字セットを指定します。このオプションの値は、ロケール・ファイル内のエントリに対応するロケール名でなければなりません。-J を指定しない場合、プリコンパイラはソース・ファイルがプリコンパイラのデフォルトの文字セットを使用しているものと解釈します。

デフォルトとして使用する文字セットを決定するために、プリコンパイラはロケール名を調べます。CS-Library は、次の順序で検索します。

1 LC_ALL

2 LANG

これらのロケール値がいずれも定義されていない場合、CS-Library は「デフォルト」のロケール名を使用します。

プリコンパイラは *locales.dat* ファイルでロケール名を探し、そのロケール名に関連付けられている文字セットをデフォルトの文字セットとして使用します。

-K *syntax_level*
実行する構文チェックのレベルを指定します。選択肢は次のとおりです。

- none
- syntax
- semantic

syntax または semantic を使用する場合は、Embedded SQL が Adaptive Server に接続できるように、-U、-P、-S、-D も指定する必要があります。

このオプションを使用しない場合、プリコンパイラはサーバに接続しないか、ターゲット・ファイルを生成するために必要な部分を除く入力ファイルの SQL 構文チェックを実行します。

-L *listing_file_name*
1 つまたは複数のリスティング・ファイルを生成します。リスティング・ファイルは、行番号が付けられた各行の後に、エラーがある場合は該当するエラー・メッセージが続く入力ファイルの 1 つのバージョンです。複数の入力ファイルがある場合は -L を使用できますが、引数を指定することはできません。

複数の入力ファイルがある場合、または引数を指定しない場合は、デフォルトのリスティング・ファイル名は、*.lis* 拡張子を付加した (または入力ファイル名の拡張子をこの拡張子に置き換えた) 入力ファイル名になります。

このオプションを指定しない場合は、リスティング・ファイルは生成されません。

-M
セキュリティ機能をオンにして、セキュリティ・ラベルを B1 に設定します。

-N *interface_file_name*
プリコンパイラに対して設定ファイルの名前 (*sql.ini*) を指定します。

-O *target_file_name*
ターゲット・ファイルとなる出力ファイルの名前を指定します。複数の入力ファイルがある場合は、このオプションを使用できません (デフォルトのターゲット・ファイル名が割り当てられます)。このオプションを使用しない場合、デフォルトのターゲット・ファイル名は、*.cbl* 拡張子を付加した (または入力ファイル名の拡張子をこの拡張子に置き換えた) 入力ファイル名になります。

-P *password*
プリコンパイル時に SQL 構文チェックを実行するための Adaptive Server パスワードを指定します。-P を引数なしで使用するか、引数としてキーワード NULL を指定すると、null (“ ”) パスワードが指定されます。-P を使用せずに -Uuser_id オプションを使用した場合、プリコンパイラはパスワードの入力を求めるプロンプトを表示します。このオプションは -G フラグとともに使用してください。

-S *server_name*

プリコンパイル時に SQL 構文チェックを実行する場合の Adaptive Server の名前を指定します。このオプションを使用しない場合は、DSQUERY 環境変数からデフォルトの Adaptive Server 名が取得されます。DSQUERY が設定されていない場合には、SYBASE がサーバの名前として使用されます。

-T *tag_id* (-G とともに使用)

生成されるストアド・プロシージャ・グループ名の最後に付加するタグ ID (3 文字以内) を指定します。

たとえば、-Tdbg をコマンドの一部として入力した場合、生成されるストアド・プロシージャには、タグ ID として *dbg* が付加された入力ファイルの名前 (*program_dbg;1*、*program_dbg;2*, など) が割り当てられます。

プログラマはタグ ID を使用することによって、使用中の可能性がある生成済みの既存のストアド・プロシージャに影響を与えずに、既存のアプリケーションに対する変更をテストできます。

このオプションを使用しない場合は、ストアド・プロシージャ名にタグ ID は追加されません。

-U *user_id*

Adaptive Server のユーザ ID を指定します。

このオプションを使用すると、プリコンパイル時に SQL 構文を検査できます。このオプションを使用すると、プリコンパイラは解析だけを目的として SQL 文をサーバに渡します。サーバが構文エラーを検出すると、エラーがレポートされてコードは生成されません。-P*password* を使用していない場合は、パスワードの入力を求めるプロンプトが表示されます。

-K、-P、-S、-D も参照してください。

-V *version_number*

Client-Library のバージョン番号を指定します。バージョン番号は、cobpub.cbl の値のいずれかと一致する必要があります。このオプションを使用しない場合、デフォルトはプリコンパイラで使用できる Client-Library の最新バージョン (Open Client/Open Server バージョン 15.5 の場合は CS_CURRENT_VERSION) になります。

-Z *language_locale_name*

プリコンパイラがメッセージに使用する言語と文字セットを指定します。-Z を指定しない場合、プリコンパイラはそのデフォルトの言語と文字セットをメッセージに使用します。

プリコンパイラは、メッセージ用のデフォルトとして使用する言語と文字セットを次のようにして決定します。

1　ロケール名を探します。CS-Library は、次の順序で検索します。

   a　LC_ALL

   b　LANG

   これらのロケール値がいずれも定義されていない場合、CS-Library は「デフォルト」のロケール名を使用します。

2　*locales.dat* ファイル内でロケール名を探して、そのロケール名に関連付けられた言語と文字セットを確認します。

3　手順 2 で調べた言語と文字セットに対応する、ローカライズされたメッセージと文字セットの情報をロードします。

@options_file

上記のコマンド・ライン引数のいずれかを含んでいるファイルを指定するために使用します。プリコンパイラは、すでに指定されている引数に加えてこのファイルに含まれている引数を読み込みます。*@options_file* で指定するファイル内にプリコンパイルするファイルの名前が含まれている場合は、この引数はコマンド・ラインの最後に置いてください。

-a

トランザクション間で、カーソルをオープンしたままにできるようにします。このオプションを使用しない場合、カーソルは set close on endtran on が有効な場合と同様に動作します。これは ANSI 準拠の動作です。詳細については、『ASE リファレンス・マニュアル』を参照してください。

-b

fetch 文で一般的に使用されるホスト変数アドレスの再バインドを無効にします。このオプションを使用しない場合は、Embedded SQL/C プログラム内で別の方法で指定しないかぎり、fetch 文が出現するたびに再バインドが行われます。

-b オプションは、Embedded SQL プリコンパイラのバージョンによって次のように異なります。

- 11.1 以降のバージョンの cobpre では、-b オプションを使用して宣言がプリコンパイルされているカーソルのすべての fetch 文に norebind 属性が適用されます。
- 10.0 以前のバージョンの cobpre では、カーソルが宣言された場所に関係なく、-b オプションを使用してプリコンパイルされた各 Embedded SQL ソース・ファイル内のすべての fetch 文に norebind 属性が適用されます。

-c

ct_debug に対する呼び出しを生成することによって Client-Library のデバッグ機能をオンにします。

このオプションは、アプリケーションの開発時には役立ちますが、アプリケーションを最終的に配布するときにはオフにしてください。このオプションを適切に機能させるには、*%SYBASE%¥%SYBASE_OCS%¥devdll* ディレクトリにあるライブラリと、*%SYBASE%¥%SYBASE_OCS¥devdll* ディレクトリにある DLL にアプリケーションをリンクして実行する必要があります。

-d

区切り識別子 (二重引用符で囲まれた識別子) をオフにし、SQL 文内の引用符で囲まれた文字列を文字リテラルとして扱えるようにします。

-e

exec sql connect 文を処理するときに、外部設定ファイルを使用して接続を設定するよう Client-Library に指示します。-x オプションと、『Open Client Client-Library/C リファレンス・マニュアル』の CS_CONFIG_BY_SERVERNAME プロパティも参照してください。

このオプションを使用しない場合、プリコンパイラは Client-Library の関数呼び出しを生成して接続を設定します。外部設定ファイルの詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

-f

ANSI FIPS 準拠の検査を行うための FIPS フラガをオンにします。

-l

#line ディレクティブを生成しないようにします。

-m

アプリケーションを Sybase のオートコミット・モードで実行します。このモードではトランザクションが連鎖しません。明示的な begin トランザクションと end トランザクションが必要となります。これらがない場合は、各文が即座にコミットされます。このオプションを指定しない場合、アプリケーションは ANSI 形式の連鎖トランザクション・モードで実行されます。

-r

繰り返し読み出しを無効にします。このオプションを使用していない場合、connect 文の最中に実行される、set transaction isolation level 3 文が生成されます。デフォルトの独立性レベルは 1 です。

-s

静的関数宣言をインクルードします。

-u

ANSI 形式のバインドを無効にします。

-v

(プリコンパイルを実行せずに) プリコンパイラのバージョン情報だけを表示します。

-w
警告メッセージの表示をオフにします。

-x
外部設定ファイルを使用します。『Open Client Client-Library/C リファレンス・マニュアル』に記載されている CS_EXTERNAL_CONFIG プロパティと、『Open Client Embedded SQL/C プログラマーズ・ガイド』に記載されている INITIALIZE_APPLICATION 文の説明を参照してください。

-y
S_TEXT データ型と CS_IMAGE データ型を入力ホスト変数として使用できるようにします。実行時に、データはサーバに送信される文字列に直接挿入されます。サポートされるのは静的 SQL 文だけです。動的 SQL の入力パラメータとして、text と image を使用することはできません。引数をコマンド文字列に置き換えるのは、-y コマンド・ライン・オプションを使用する場合だけです。

filename[.ext]
ESQL/C ソース・プログラムの入力ファイル名を指定します。ファイル名の形式と長さは、規則に違反しないかぎり、どのようなものでもかまいません。

例

例 1 プリコンパイラを実行します (ANSI 準拠)。

```
cobpre program.pco
```

例 2 生成されたストアド・プロシージャと FIPS フラグを使用して、プリコンパイラを実行します (ANSI 準拠)。

```
cobpre -G -f program1.pco
```

例 3 トランザクション間でカーソルがオープンしたままの状態で、入力ファイルに対してプリコンパイラを実行します (ANSI 非準拠)。

```
cobpre -a program1.pco
```

例 4 プリコンパイラのバージョン情報だけを表示します。

```
cobpre -v
```

例 5 最高レベルの SQL チェックを指定して、プリコンパイラを実行します。

```
cobpre -KSEMANTIC -Uuser_id -Ppassword -Sserver_name
-Dpubs2 example1.pco
```

使用法

- cobpre| コマンドのデフォルトは、ANSI 標準の動作に対して設定されます。
- -a、-c、-f、-m、-r、-V オプションは connect 文だけに影響します。ソース・ファイルに connect 文が含まれていない場合、または -e か -x を使用する場合は、これらのオプションは影響しません。

- ターゲット・ファイルデフォルトのターゲット・ファイル名は、*.cbl* 拡張子 (Micro Focus COBOL の場合) が付加された (または入力ファイル名の拡張子をこれらの拡張子に置き換えた) 入力ファイル名です。入力ファイルが 1 つだけの場合は、-O target_file_name オプションを使用してターゲット・ファイル名を指定できます。複数の入力ファイルがある場合は、デフォルトのターゲット・ファイルは、*first_input_file.cbl*、*second_input_file.cbl* などになります。
- オプションは、引数の前にスペースがあってもなくてもかまいません。たとえば、次のどちらでも問題ありません。
  - -Tdbg
  - -T dbg
- プリコンパイラは複数の入力ファイルを処理できます。-O *target_file_name* オプションを使用しなくてもかまいませんが、その場合はデフォルトのターゲット・ファイル名を使用する必要があります (上記の「ターゲット・ファイル」の説明を参照)。-G *isql_file_name* を使用する場合は、引数を指定できません。デフォルトの isql ファイル名は、*first_input_file.sql*、*second_input_file.sql* などです。-L *listing_file_name* を使用する場合は、引数を指定することはできません。デフォルトのリスティング・ファイル名は、*first_input_file.lis*、*second_input_file.lis* などです。
- デフォルトでは、cobpre はインジケータ変数の ANSI 形式のバインド (CS_ANSI_BINDS) を有効にする ct_options に対する呼び出しを生成します。null 入力可能なホスト変数を表すインジケータ変数 (*columns*) が使用できない場合、Client-Library は致命的な実行時エラーを生成し、使用中のアプリケーションをアボートします。これらの問題は、cobpre とともに -u を使用することで回避できます。*ocs.cfg* ファイルで CS_ANSI_BINDS を cs_false に設定して、ANSI 形式のバインドを無効にすることもできます。

アプリケーションの開発

この項では、Embedded SQL アプリケーションの開発で最も一般的に使用される手順について説明します。この手順は、稼働条件に合うように適応させることが必要な場合もあります。これらの手順は、DOS コマンド・プロンプトから実行してください。

1 構文チェックとデバッグを行うために、-c、-Ddatabase_name、-P[password]、-Sserver_name、-K[SYNTAX| SEMANTIC]、-Uuser_id の各オプションを使用して、プリコンパイラを実行します。
-G[isql_file_name] は使用しないでください。プログラムのコンパイルとリンクを実行して、構文が正しいかどうか確認します。

2　必要な修正をすべて行います。-Ddatabase_name、-G[isql_file_name]、-Ttag_id の各オプションを使用してプリコンパイラを実行し、テスト・プログラム用のタグ ID を持つストアド・プロシージャを生成します。テスト・プログラムをコンパイルしてリンクします。次のコマンドを使用して、ストアド・プロシージャをロードします。

```
isql -Ppassword -Sserver_name -Uuser_id -iisql_file_name
```

プログラム上でテストを実行します。

3　修正版のプログラムに対して、-Ddatabase_name と -G[isql_file_name] の各オプションを使用して (-T オプションは使用しない) プリコンパイラを実行します。プログラムをコンパイルしてリンクします。次のコマンドを使用して、ストアド・プロシージャをロードします。

```
isql -Ppassword -Sserver_name -Uuser_id -iisql_file_name
```

これで、最終的な配布用プログラムを実行する準備が完了しました。

プリコンパイラがサーバの名前を確認する方法

プリコンパイル時に Adaptive Server に接続することによって、プリコンパイル時に追加で構文チェックを実行できます。プリコンパイラは、次の 3 つの方法のいずれかを使用してサーバの名前を調べます。

- cpre または cobpre コマンド・ラインで -S オプションを使用する
- DSQUERY 変数を設定する
- デフォルト値 “SYBASE” を使用する

-S オプションは、DSQUERY によって設定された値を上書きします。

プリコンパイル・コマンド・ラインでサーバを選択するには、次の構文を使用します。

```
cobpre -Usa -P -Sserver_name
```

別の方法として、接続呼び出しまたは接続文のサーバ名を無視することもできます。この場合、*server_name* は DSQUERY 環境変数のランタイム値から値を取得します。アプリケーション・ユーザが DSQUERY を設定していない場合、サーバ名のランタイム値はデフォルトの “SYBASE” になります。DSQUERY の詳細については、『Open Client/Server 設定ガイド Windows 版』を参照してください。

cobpre のデフォルト

cpre と cobpre のオプションとデフォルトのリストについては、表 A-6 を参照してください。

defncopy

説明

指定されたビュー、ルール、デフォルト、トリガ、プロシージャの定義を、データベースからオペレーティング・システム・ファイルに、またはオペレーティング・システム・ファイルからデータベースにコピーします。このユーティリティは *%SYBASE%¥%SYBASE_OCS%¥bin* にあります。

注意 defncopy では、Report Workbench™ を使用して作成したテーブル定義またはレポートをコピーすることはできません。

構文

```
defncopy
    [-a display_charset]
    [-I interfaces_file]
    [-J [client_charset]]
    [-K keytab_file]
    [-P password]
    [-R remote_server_principal]
    [-S [server_name]]
    [-U user_name]
    [-v]
    [-V [security_options]]
    [-X]
    [-z language]
    [-Z security_mechanism]
    {in file_name database_name |  out file_name database_name
    [owner.]object_name [[owner.]object_name...] }
```

パラメータ

-a *display_charset*

defncopy が実行されているマシンの文字セットと異なる文字セットを使用する端末から、defncopy を実行します。-a を -J とともに使用して、変換に必要な文字セット変換ファイル (*.xlt* ファイル) を指定します。-J を指定しないで -a を使用するのは、クライアントの文字セットがデフォルトの文字セットと同じである場合だけです。

注意 ascii_7 文字セットは、すべての文字セットと互換性があります。Adaptive Server の文字セットとクライアントの文字セットのどちらかが ascii_7 である場合は、すべての 7 ビット ASCII 文字を変更することなくクライアントとサーバの間で渡すことができます。その他の文字セットを使用している場合は、変換エラーが発生します。文字セット変換に関する問題の詳細については、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。

-I *interfaces_file*

Adaptive Server に接続するときに検索する interfaces ファイルの名前とロケーションを指定します。-I を指定しない場合、defncopy は *%SYBASE%¥ini* にある interfaces ファイル (*sql.ini*) を探します。

-J *client_charset*

クライアントで使用する文字セットを指定します。フィルタによって、*client_charset* と Adaptive Server の文字セット間で入力が変換されます。

-J *client_charset* は、クライアントの文字セットである *client_charset* とサーバの文字セット間の変換を Adaptive Server に要求します。

-J に引数を指定しない場合、文字セット変換は NULL に設定されます。この場合、変換は行われません。クライアントとサーバが同じ文字セットを使用する場合に、このパラメータを使用してください。

-J を省略すると、文字セットはプラットフォームのデフォルトに設定されます。デフォルトの文字セットは、クライアントが使用する文字セットと同じであるとはかぎりません。

-K *keytab_file*

DCE セキュリティでのみ使用します。-U パラメータで指定されたユーザ名のセキュリティ・キーを含む DCE *keytab* ファイルを指定します。*Keytab* ファイルは、DCE dcecp ユーティリティを使用して作成できます。詳細については、DCE のマニュアルを参照してください。

-K パラメータを指定しない場合、defncopy のユーザは、-U パラメータで指定されたユーザ名と同じユーザ名を使用して DCE にログインする必要があります。

-P *password*

パスワードを指定できるようにします。-P を指定しない場合、defncopy はパスワードの入力を求めるプロンプトを表示します。-V を指定すると、このオプションは無視されます。

-R *remote_server_principal*

リモート・サーバのプリンシパル名を指定します。デフォルトでは、サーバのプリンシパル名はサーバのネットワーク名 (-S パラメータまたは DSQUERY 環境変数で指定) と一致します。サーバのプリンシパル名とネットワーク名が異なる場合は、-R パラメータを使用してください。

-S *server_name*

接続先の Adaptive Server の名前を指定します。引数なしで -S を指定した場合、defncopy は SYBASE という名前のサーバを探します。-S を指定しない場合、defncopy は DSQUERY 環境変数で指定されたサーバを使用します。

-U *user_name*

ログイン名を指定できるようにします。ログイン名では大文字と小文字が区別されます。*username* を指定しない場合、defncopy は現在のユーザのオペレーティング・システムのログイン名を使用します。

-v

defncopy のバージョン番号と版権メッセージを表示して、オペレーティング・システムに戻ります。

-V *security_options*

ネットワーク・ベースのユーザ認証を指定します。ユーザは defncopy を実行する前に、ネットワークのセキュリティ・システムにログインする必要があります。この場合、ユーザは -U パラメータでネットワーク・ユーザ名を指定します。-P パラメータで指定されたパスワードは無視されます。

-V の後に *security_options* 文字列を指定することによって、追加のセキュリティ・サービスを有効にできます。指定できる文字は次のとおりです。

- c − データ機密性サービスを有効にする。
- i − データ整合性サービスを有効にする。
- m − 接続を確立するための相互認証を有効にする。
- o − データ・オリジン・スタンピング・サービスを有効にする。
- q − 順序不整合の検出を有効にする。
- r − データ・リプレイの検出を有効にする。

-X

サーバへの現在の接続で、アプリケーションがクライアント側のパスワード暗号化を使用してログインを開始することを指定します。defncopy (クライアント) は、パスワードの暗号化が必要であることをサーバに通知します。サーバは暗号化キーを送り返し、defncopy はそれを使用してパスワードを暗号化します。サーバはパスワードを受け取ると、そのキーを使用してパスワードの認証を行います。

defncopy が失敗した場合、パスワードを含むコア・ファイルが作成されます。暗号化オプションを使用していない場合、パスワードは、コア・ファイルにプレーン・テキストで表示されます。暗号化オプションを使用した場合、パスワードは表示されません。

-z *language*

サーバが defncopy のプロンプトとメッセージの表示に使用する代替言語の公式名です。-z フラグが指定されていない場合、defncopy はサーバのデフォルト言語を使用します。

インストール時に言語を Adaptive Server に追加します。インストール後でも、langinst ユーティリティまたは sp_addlanguage ストアド・プロシージャを使用して追加できます。

-Z *security_mechanism*

接続で使用するセキュリティ・メカニズムの名前を指定します。

セキュリティ・メカニズムの名前は、*%SYBASE%¥%SYBASE_OCS%¥ini* ディレクトリ内にある *libtcl.cfg* 設定ファイルに定義されています。*security_mechanism* の名前が指定されていない場合は、デフォルトのメカニズムが使用されます。詳細については、『Open Client/Server 設定ガイド Windows 版』の *libtcl.cfg* ファイルの説明を参照してください。

in | out
定義をコピーする方向を指定します。

file_name
定義コピーの送信元または送信先であるオペレーティング・システム・ファイルの名前を指定します。コピー・アウトを行うと、既存のファイルはすべて上書きされます。

database_name
定義のコピー先またはコピー元であるデータベースの名前を指定します。

object_name
defncopy がコピー・アウトするデータベース・オブジェクトの名前を指定します。定義をコピー・インするときは、*object_name* を使用しないでください。

owner
コピーするテーブルをユーザまたはデータベース所有者が所有している場合は、このパラメータの指定はオプションです。所有者を指定しない場合、defncopy はまずユーザが所有する該当の名前のテーブルを探します。次に、データベース所有者が所有するテーブルを探します。別のユーザがテーブルを所有している場合は、所有者の名前を指定する必要があります。指定しないと、コマンドは失敗します。

例

例 1 *new_proc* ファイルから、MERCURY サーバ上の stagedb データベースに定義をコピーします。MERCURY との接続は、ユーザ名 “sa”、パスワード NULL を使用して確立されます。

```
defncopy -Usa -P -SMERCURY in new_proc stagedb
```

例 2 Sybase サーバ上の employees データベースから *dc.out* ファイルに、sp_calccomp オブジェクトと sp_vacation オブジェクトの定義をコピーします。メッセージとプロンプトはフランス語で表示されます。ユーザは、パスワードを入力するように要求されます。

```
defncopy -S -z french out dc.out employees sp_calccomp sp_vacation
```

使用法

- defncopy プログラムは、オペレーティング・システムから直接呼び出します。defncopy では、ビュー、ルール、デフォルト、トリガ、またはプロシージャの各定義 (create 文) をデータベースからオペレーティング・システム・ファイルに非対話型操作でコピー・アウトできます。または、指定されたファイルからすべての定義をコピーできます。
- コピー・インするには、コピー対象のオブジェクトのタイプに対する適切な create パーミッションが必要です。コピー・インされたオブジェクトは、コピーを実行したユーザの所有物となります。ユーザの代わりに定義をコピー・インするシステム管理者は、そのユーザとしてログインして、再構築したデータベース・オブジェクトへの適切なアクセス権をユーザに与える必要があります。

- in *filename* または out *filename* とデータベース名は必須です。名前を明確に指定してください。コピー・アウトする場合は、オブジェクト名とその所有者の両方を表すファイル名を使用してください。
- defncopy は、コピー・アウトする各定義を次のようなコメントで終了します。

```
/* ### DEFNCOPY: END OF DEFINITION  */
```

  defncopy を使用してデータベースにコピーするオペレーティング・システム・ファイル内の定義をアセンブルする場合、各定義は “END OF DEFINITION” という文字列を使用して終了する必要があります。
- defncopy に対して指定した値に、シェルにとって特別な意味のある文字が含まれている場合は、それらの値を引用符で囲みます。

警告！ create 文の前に 100 文字を超える長いコメントがあると、defncopy が失敗することがあります。

パーミッション

- 定義をコピー・アウトするには、sysobjects テーブルと syscomments テーブルに対する select パーミッションが必要です。オブジェクト自体に対するパーミッションは必要ありません。
- システム・セキュリティ担当者が sp_configure を使用して allow select on syscomment.text column パラメータを再設定した場合、ユーザは syscomments テーブルの text カラムに対する select パーミッションを持つことはできません。この再設定によって、select パーミッションはオブジェクト所有者とシステム管理者に制限されます。使用しているプラットフォームの Adaptive Server Enterprise の『インストール・ガイド』と『設定ガイド』に記載されているように、この制限は Adaptive Server を「評価済み設定」で実行するために必要となります。この場合、オブジェクト所有者またはシステム管理者は、defncopy を実行して定義をコピー・アウトします。

注意 テキストが暗号化されている場合、必要なパーミッションをすべて持っていても、表示されないことがあります。詳細については、Adaptive Server Enterprise の『Transact-SQL ユーザーズ・ガイド』の「ソース・テキストの検証および暗号化」を参照してください。

- コピー・インするには、コピー対象のオブジェクトのタイプに対する適切な create パーミッションが必要です。コピー・インされたオブジェクトは、コピーを実行したユーザの所有物となります。ユーザの代わりに定義をコピー・インするシステム管理者は、そのユーザとしてログインして、再構築したデータベース・オブジェクトへの適切なアクセス権をユーザに与える必要があります。

isql

説明

isql は、Adaptive Server の対話型 SQL パーサです。このユーティリティは *%SYBASE%¥%SYBASE_OCS%¥bin* にあります。

構文

isql [-b] [-e] [-F] [-n] [-p] [-v] [-X] [-Y] [-Q]
[-a *display_charset*]
[-A *packet_size*]
[-c *cmdend*]
[-D *database*]
[-E *editor*]
[-h *header*]
[-H *hostname*]
[-i *inputfile*]
[-I *interfaces_file*]
[-J *client_charset*]
[-K *keytab_file*]
[-l *login_timeout*]
[-m *errorlevel*]
[-o *outputfile*]
[-P *password*]
[-R *remote_server_principal*]
[-s *col_separator*]
[-S *server_name*]
[-t *timeout*]
[-U *username*]
[-V [*security_options*]]
[-w *column_width*]
[-W]
[-y *alternate_home_directory*]
[-z *localename*]
[-Z *security_mechanism*]
[--appname "*application_name*"]
[--conceal [':?' | '*wildcard*']]
[--help]
[--history [p]*history_length* [--history_file *history_filename*]]
[--retserverror]

パラメータ

-a *display_charset*

isql を実行しているマシンの文字セットと異なる文字セットを使用する端末から、isql を実行できるようにします。-a を -J とともに使用して、変換に必要な文字セット変換ファイル (*.xlt* ファイル) を指定します。-J を指定しないで -a を使用するのは、クライアントの文字セットがデフォルトの文字セットと同じである場合だけです。

注意　ascii_7 文字セットは、すべての文字セットと互換性があります。Adaptive Server の文字セットとクライアントの文字セットのどちらかが ascii_7 である場合は、すべての 7 ビット ASCII 文字を変更することなくクライアントとサーバの間で渡すことができます。その他の文字セットを使用している場合は、変換エラーが発生します。文字セット変換に関する問題の詳細については、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。

-A *packet_size*

この isql セッションに使用するネットワーク・パケット・サイズを指定します。たとえば、この isql セッションのパケット・サイズを 4096 バイトに設定するには、次のように入力します。

```
isql -A 4096
```

ネットワーク・パケット・サイズをチェックするには、次のように入力します。

```
select * from sysprocesses
```

値は *network_pktsz* という見出しの下に表示されます。

packet_size は、default network packet size 設定変数と maximum network packet size 設定変数の間の値であり、512 の倍数であることが必要です。

readtext や writetext などの I/O を集中的に使用するオペレーションを実行する場合は、パケット・サイズをデフォルトより大きな値に設定します。

Adaptive Server のパケット・サイズを設定または変更しても、リモート・プロシージャ・コールのパケット・サイズには影響しません。

-b

テーブル・ヘッダの出力表示を無効にします。

-c *cmdend*

コマンド・ターミネータを再設定します。デフォルトでは、行に “go” と入力するだけで、各コマンドを終了し、Adaptive Server に送信できます。コマンド・ターミネータを再設定する場合は、SQL の予約語や制御文字を使用しないでください。“?”、“( )”、“[ ]”、“$” などのシェル・メタ文字は、必ずエスケープしてください。

-D *database*

isql セッションを開始するデータベースを選択します。

-e

入力内容をエコーします。

-E *editor*

デフォルト・エディタ (edit など) 以外のエディタを指定します。エディタを呼び出すには、isql で行の最初の単語としてエディタ名を入力します。

-F

FIPS フラグを有効にします。-F パラメータを指定した場合、サーバは非標準 SQL のコマンドを検出するとメッセージを返します。このオプションは、SQL 拡張機能を無効にするわけではありません。ANSI SQL 以外のコマンドを発行すると、処理は完了します。

-h *header*

カラム見出しから次のカラム見出しまでの間に出力されるローの数を指定します。デフォルトでは、クエリ結果のセットごとに 1 回だけ見出しが出力されます。

-H *hostname*

クライアント・ホスト名を設定します。

-i *inputfile*

isql への入力に使用するオペレーティング・システム・ファイルの名前を指定します。このファイルには、コマンド・ターミネータ (デフォルトでは “go”) が含まれている必要があります。

- パラメータを次のように指定すると、< *inputfile* を指定した場合と同じになります。

  ```
  -i inputfile
  ```

- -i を使用し、コマンド・ラインにパスワードを指定しない場合、isql はパスワードの入力を求めるプロンプトを表示します。
- <*inputfile* を使用し、コマンド・ラインにパスワードを指定しない場合は、入力ファイルの最初の行にパスワードを指定してください。

-I *interfaces_file*

Adaptive Server に接続するときに検索する interfaces ファイルの名前とロケーションを指定します。-I を指定しない場合、isql は *%SYBASE%¥ini* ディレクトリにある interfaces ファイル (*sql.ini*) を探します。

-J *client_charset*

クライアントで使用する文字セットを指定します。-J *client_charset* は、クライアントで使用する文字セットである *client_charset* とサーバの文字セット間の変換を Adaptive Server に要求します。フィルタによって、*client_charset* と Adaptive Server の文字セット間で入力が変換されます。

-J に引数を指定しない場合、文字セット変換は NULL に設定されます。この場合、変換は行われません。クライアントとサーバが同じ文字セットを使用する場合に、このパラメータを使用してください。

-J を省略すると、文字セットはプラットフォームのデフォルトに設定されます。デフォルトの文字セットは、クライアントが使用する文字セットと同じであるとはかぎりません。

-K *keytab_file*

DCE セキュリティでのみ使用できます。-U オプションで指定されたユーザ名のセキュリティ・キーを含む DCE *keytab* ファイルを指定します。Keytab ファイルは、DCE dcecp ユーティリティを使用して作成できます。詳細については、DCE のマニュアルを参照してください。

-K オプションを指定しない場合、isql のユーザは -U オプションで指定したユーザ名と同じユーザ名を使用して DCE にログインする必要があります。

-l *login_timeout*

Adaptive Server に接続する場合の最大タイムアウト値を指定します。デフォルトは 60 秒です。この値は、サーバがログインの要求に応答するのを isql が待つ時間に対してのみ影響します。コマンド処理のタイムアウト時間を指定するには、-t *timeout* パラメータを使用します。

-m *errorlevel*

エラー・メッセージの表示をカスタマイズします。指定の重大度レベル以上のエラーの場合には、メッセージ番号、ステータス、エラー・レベルを表示し、エラー・テキストは表示しません。指定した重大度より低いレベルのエラーでは、何も表示されません。

-n

-e とともに使用した場合、出力ファイルにエコーされた入力行から、行番号とプロンプト記号 (>) を削除します。

-o *outputfile*

isql からの出力を保管するオペレーティング・システム・ファイルの名前を指定します。パラメータを -o *outputfile* と指定するのは、>*outputfile* と指定するのと同じです。

-p

パフォーマンスの統計値を出力します。

-P *password*

Adaptive Server のパスワードを指定します。-V を指定すると、このオプションは無視されます。パスワードは大文字と小文字が区別され、6 ～ 30 文字の範囲で指定できます。パスワードが NULL の場合は、パスワードを指定せずに -P を使用します。

-Q

クライアントにフェールオーバ (HA) プロパティを提供します。詳細については、Adaptive Server Enterprise の『高可用性システムにおける Sybase フェールオーバの使用』を参照してください。

-R *remote_server_principal*

セキュリティ・メカニズムに定義されたサーバのプリンシパル名を指定します。デフォルトでは、サーバのプリンシパル名はサーバのネットワーク名 (-S オプションまたは DSQUERY 環境変数で指定) と一致します。サーバのプリンシパル名とネットワーク名が異なる場合に、-R を使用します。

-s *col_separator*

カラム・セパレータ文字をリセットします。デフォルト・カラム・セパレータ文字はブランクです。オペレーティング・システムに対して特別な意味を持つ文字 (“|”、“;”、“&”、“<”、“>” など) を使用するには、これらの文字を引用符で囲むか、前に円記号を付けます。

カラム・セパレータは、各ローの各カラムの先頭と末尾に表示されます。

-S *server_name*

接続先の Adaptive Server の名前を指定します。isql は、この名前のエントリを interfaces ファイルで探します。-S を引数なしで指定した場合、isql は SYBASE という名前のサーバを探します。-S を指定しない場合には、isql は DSQUERY 環境変数によって指定されたサーバを探します。

-t *timeout*

SQL コマンドがタイムアウトするまでの秒数を指定します。タイムアウト値の指定がないと、コマンドは永久に実行を続けます。これは、接続時間ではなく、isql 内から発行されたコマンドに影響します。isql にログインするためのデフォルトのタイムアウトは 60 秒です。

-U *username*

ログイン名を指定します。ログイン名では大文字と小文字が区別されます。

-V *security_options*

ネットワーク・ベースのユーザ認証を指定します。このオプションを使用する場合、ユーザは isql を実行する前に、ネットワークのセキュリティ・システムにログインする必要があります。この場合、ユーザは -U オプションでネットワーク・ユーザ名を指定します。-P オプションで指定されたパスワードは無視されます。

-V の後に *security_options* 文字列を指定することによって、追加のセキュリティ・サービスを有効にできます。指定できる文字は次のとおりです。

- c − データ機密性サービスを有効にする。
- d − クレデンシャル委任を有効にし、クライアント・クレデンシャルをゲートウェイ・アプリケーションに転送する。
- i − データ整合性サービスを有効にする。
- m − 接続を確立するための相互認証を有効にする。
- o − データ・オリジン・スタンピング・サービスを有効にする。
- q − 順序不整合の検出を有効にする。
- r − データ・リプレイの検出を有効にする。

-v

isql のバージョンと版権メッセージを表示して、終了します。

-w *column_width*

出力画面の幅を設定します。デフォルトでは、80 文字です。出力行が画面幅いっぱいになった場合は、複数の行に分割されます。

-W

isql が接続しようとしているサーバが通常のパスワード暗号化と拡張パスワード暗号化のどちらもサポートしていない場合、プレーン・テキスト形式のパスワードを使用した接続再試行を無効にすることを指定します。このオプションを使用すると、CS_SEC_NON_ENCRYPTION_RETRY 接続プロパティが CS_FALSE に設定され、接続の再試行時にプレーン・テキスト形式の (暗号化されていない) パスワードは使用されなくなります。

注意　このリリースでは、-W オプションと CS_SEC_NON_ENCRYPTION_RETRY プロパティは無視されます。

-X

クライアント側のパスワード暗号化を使用して、サーバへのログイン接続を開始します。isql (クライアント) は、パスワードの暗号化が必要であることをサーバに通知します。サーバは、isql がパスワードの暗号化に使用する暗号化キーを返送し、パスワードを受け取ると、そのキーを使用してパスワードを認証します。

このオプションでは、サーバでの接続プロパティの設定に応じて、通常のパスワード暗号化が使用される場合もあれば、拡張パスワード暗号化が使用される場合もあります。CS_SEC_ENCRYPTION が CS_TRUE に設定されている場合は、通常のパスワード暗号化が使用されます。CS_SEC_EXTENDED_ENCRYPTION が CS_TRUE に設定されている場合は、拡張パスワード暗号化が使用されます。CS_SEC_ENCRYPTION と CS_SEC_EXTENDED_ENCRYPTION のどちらも CS_TRUE に設定されている場合は、拡張パスワード暗号化が優先的に使用されます。

isql が失敗した場合、パスワードを含むコア・ファイルが作成されます。暗号化オプションを使用していない場合、パスワードは、コア・ファイルにプレーン・テキストで表示されます。暗号化オプションを使用した場合、パスワードは表示されません。

-y *alternate_home_directory*

代替の Sybase ホーム・ディレクトリを設定します。

-Y

連鎖トランザクションを使用するように Adaptive Server に指示します。

-z *localename*

isql のプロンプトとメッセージの表示に使用する代替言語の公式名です。-z を指定しない場合、isql はサーバのデフォルト言語を使用します。インストール時に言語を Adaptive Server に追加します。インストール後でも、langinst ユーティリティまたは sp_addlanguage ストアド・プロシージャを使用して追加できます。

-Z *security_mechanism*

接続で使用するセキュリティ・メカニズムの名前を指定します。

セキュリティ・メカニズムの名前は、*%SYBASE%¥%SYBASE_OCS%¥ini* ディレクトリ内にある *libtcl.cfg* 設定ファイルに定義されています。*security_mechanism* の名前が指定されていない場合は、デフォルトのメカニズムが使用されます。詳細については、『Open Client/Server 設定ガイド Windows 版』の *libtcl.cfg* ファイルの説明を参照してください。

--appname "*application_name*"

デフォルトのアプリケーション名である *isql* を isql クライアント・アプリケーション名に変更できます。これにより、次のことが容易になります。

- クライアント・アプリケーション名に基づいた、クライアント受信接続に関する Adaptive Server クラスタのルート指定ルールのテスト
- *%SYBASE%¥%SYBASE_OCS%¥ini¥ocs.cfg* にある isql の代替設定の切り替え (デバッグ・セッションと通常セッションの切り替えなど)
- Adaptive Server 内から特定の isql セッションを開始したスクリプトの識別

application_name は、クライアント・アプリケーション名です。クライアント・アプリケーション名は、ホスト・サーバへの接続後に、sysprocesses.program_name から取得できます。

application_name の最大長は 30 文字です。アプリケーション名に円記号エスケープ文字を使用しないスペースが含まれている場合は、アプリケーション名全体を一重引用符または二重引用符で囲む必要があります。*application_name* は、空の文字列に設定できます。

注意　*ocs.cfg* 内で CS_APPNAME プロパティを使用して、クライアント・アプリケーション名を設定することもできます。

--conceal [':?' | '*wildcard*']

isql セッション中の入力内容を隠します。--conceal オプションは、パスワードなどの機密情報を入力するときに役立ちます。

32 バイト変数である *wildcard* は、isql セッション中にユーザに入力を要求するプロンプトを表示するために、isql をトリガする文字列を指定します。isql がワイルドカードを読み込むたびに、isql はユーザ入力を受け入れるプロンプトを表示しますが、入力内容を画面にエコーすることはありません。デフォルトのワイルドカードは :? です。

注意　--conceal は、バッチ・モードでは暗黙的に無視されます。

---

isql セッションでのワイルドカードの使用方法については、「isql セッションでのプロンプト・ラベルと二重ワイルドカードの使用」(125 ページ) を参照してください。

--help

isql ユーティリティで使用できるすべてのコマンド・ライン・パラメータをリストします。リストには、各パラメータの機能の簡単な説明も含まれています。

--history [p]*history_length* [--history_file *history_filename*]

isql の起動時に、コマンド履歴ログ・ファイルの内容をロードします (ログ・ファイルが存在する場合)。デフォルトでは、コマンド履歴機能はオフになっています。この機能をアクティブにするには、--history コマンド・ライン・オプションを使用します。

p は、コマンド履歴の永続性を示します。メモリ内のコマンド履歴は、isql のシャット・ダウン時にディスクに保存されます。p オプションを使用しない場合、コマンド履歴ログは内容がメモリにロードされると削除されます。

history_length は、isql がコマンド履歴ログに格納できるコマンドの数です。--history を使用する場合、このパラメータは必須です。*history_length* の最大値は 1024 です。これより大きい値を指定すると、isql は 1024 に暗黙的にトランケートします。

--history_file *history_filename* は、isql がコマンド履歴ログを *history_filename* から取得しなければならないことを指定します。p を指定している場合、isql は *history_filename* を使用して、現在のセッションのコマンド履歴も格納します。*history_filename* には、ログ・ファイルの絶対パスまたは相対パスを含めることができます。相対パスは現在のディレクトリを起点とします。パスを指定しない場合、履歴ログは現在のディレクトリに保存されます。

--history_file が指定されていない場合、isql は %APPDATA%¥Sybase¥isql¥isqlCmdHistory.log にあるデフォルトのログ・ファイルを使用します。

これまでに使用したコマンドのリスト表示、再呼び出し、再発行については、「コマンド履歴の使用」(124 ページ) を参照してください。

--retserverror

重大度が 10 を超えるサーバ・エラーが発生したときに、isql を強制終了し、エラー・コードを返すようにします。この種の異常終了が発生すると、isql は実際の Adaptive Server エラー番号とともに “Msg” というラベルを stderr に書き込み、呼び出し元プログラムに値 “2” を返します。これまでと同様に、isql はサーバ・エラー・メッセージ全体を stdout に出力します。

例

例 1 これで、クエリを編集できるテキスト・ファイルの状態になります。ファイルに書き込みを行って保存すると、isql に戻ります。クエリが表示されるので、行に “go” とだけ入力して実行してください。

```
isql -Ujoe -Pabracadabra
1>select *
2>from authors
3>where city = "Oakland"
4>vi
```

例 2 reset によってクエリ・バッファがクリアされます。quit を入力すると、オペレーティング・システムに戻ります。

```
isql -U alma
Password:
1>select *
2>from authors
3>where city = "Oakland"
4>reset
5>quit
```

例 3 ストア ID 7896 の pubs2 データベースの出力に “#” を使用して、カラム・セパレータを作成します。

```
isql -Usa -P -s#
1> use pubs2
2> go
1> select * from sales where stor_id = "7896"
#stor_id#ord_num               #date                              #
#-------#----------------------#----------------------------------#
#7896   #124152                #              Aug 14 1986 12:00AM#
#7896   #234518                #              Feb 14 1991 12:00AM#

(2 rows affected)
```

例 4 MIT Kerberos のクレデンシャル委任を要求し、クライアント・クレデンシャルを MY_GATEWAY に転送します。

```
isql -Vd -SMY_GATEWAY
```

例 5 isql が重大度 10 以上のサーバ・エラーを検出すると、コマンド・プロンプトに値 “2” が返され、サーバ・エラー・メッセージ全体が stdout に出力されます。また、実際の ASE エラー番号とともに “Msg” というラベルが stderr に書き込まれます。

```
C:¥>isql -Uguest -Pguestpwd -SmyASE
--retserverror 2> isql.stderr
1> select no_column from sysobjects
2> go
Msg 207, Level 16, State 4:
Server 'myASE', Line 1:
Invalid column name 'no_column'.


C:¥>echo %ERRORLEVEL%
2
C:¥>type isql.stderr
Msg     207
C:¥>
```

例 6 --help オプションを使用すると、isql は使用可能な引数のリストが含まれた、isql ユーティリティの構文と使用方法の簡単な説明を返します。

```
C:¥>isql --help

usage: isql [option1] [option2] ... where [options] are...
-b                  Disables the display of the table headers output.
-e                  Echoes input.
-F                  Enables the FIPS flagger.
-p                  Prints performance statistics.
-n                  Removes numbering and the prompt symbol when used
                    with -e.
-v                  Prints the version number and copyright message.
-W                  Turn off extended password encryption on connection
                    retries.
-X                  Initiates the login connection to the server with
                    client-side password encryption.
-Y                  Tells the Adaptive Server to use chained transactions.
-Q                  Enables the HAFAILOVER property.
-a display_charset  Used in conjunction with -J to specify the character set
                    translation file (.xlt file) required for the conversion.
                    Use -a without -J only if the client character set is the
                    same as the default character set.
-A packet_size      Specifies the network packet size to use for this isql
                    session.
-c cmdend           Changes the command terminator.
-D database         Selects the database in which the isql session begins.
-E editor           Specifies an editor other than the default editor vi.
-h header           Specifies the number of rows to print between column
                    headings.
-H hostname         Sets the client host name.
-i inputfile        Specifies the name of the operating system file to use
                    for input to isql.
-I interfaces_file  Specifies the name and location of the interfaces file.
-J client_charset   Specifies the character set to use on the client.
-K keytab_file      Specifies the path to the keytab file used for
                    authentication in DCE.
-l login_timeout    Specifies the number of seconds to wait for the server
                    to respond to a login attempt.
-m errorlevel       Customizes the error message display.
-M labelname labelvalue
                    Used for security labels. See CS_SEC_NEGOTIATE for more
                    details.
-o outputfile       Specifies the name of an operating system file to store
the output from isql.
-P password         Specifies your Adaptive Server password.
-R remote_server_principal
                   Specifies the principal name for the server as defined to
                   the security mechanism.
-s col_separator   Resets the column separator character, which is blank by
                   default.
-S server_name     Specifies the name of the Adaptive Server to which to
```

```
                  connect.
-t timeout        Specifies the number of seconds before a SQL command times
                  out.
-U username       Specifies a login name. Login names are case sensitive.
-V [security_options]
                  Specifies network-based user authentication. Valid
                  [security_options]:
                  c - Enable data confidentiality service.
                  i - Enable data integrity service.
                  m - Enable mutual authentication for connection
                      establishment.
                  o - Enable data origin stamping service.
                  q - Enable out-of-sequence detection.
                  r - Enable data replay detection.
                  d - Requests credential delegation and forwards client
                      credentials.
-w column_width   Sets the screen width for output.
-y sybase_directory
                  Sets an alternate location for the Sybase home directory.
-z localename     Sets the official name of an alternate language to display
                  isql prompts and messages.
-Z security_mechanism

                  Specifies the name of a security mechanism to use on the
                  connection.
-x trusted.txt_file Specifies an alternate trusted.txt file location.
--retserverror    Forces isql to terminate and return a failure code when it
                  encounters a server error of severity greater than 10.
--conceal [wildcard]
                  Obfuscates input in an ISQL session. The optional wildcard
                  will be used as a prompt.
```

例 7 代替の Sybase ホーム・ディレクトリを *C:¥work¥NewSybase* に設定します。

```
C:¥>isql -yC:¥work¥NewSybase -Uuser1 -Psecret
-SMYSERVER
```

例 8 入力したパスワードを表示しないで、パスワードを変更します。この例では、プロンプト・ラベルとして “old” と “new” を使用します。

```
C:¥>isql -Uguest -Pguest -Smyase --conceal
1> sp_password
2> :? old
3> ,
4> :?:? new
5> go
old
new
Confirm new
Password correctly set.
(return status = 0)
```

例 9 入力したパスワードを表示しないで、パスワードを変更します。この例では、プロンプト・ラベルとしてデフォルトのワイルドカードを使用します。

```
C:¥>isql -Uguest -Pguest -Smyase --conceal
1> sp_password
2> :?
3> ,
4> :?:?
5> go
:?
:?
Confirm :?
Password correctly set.
(return status = 0)
```

例 10 現在のユーザの役割をアクティブ化します。この例では、カスタム・ワイルドカードを使用し、プロンプト・ラベルとして “role” と “password” を使用します。

```
C:¥>isql -UmyAccount --conceal '*'
Password:
1> set role
2> * role
3> with passwd
4> ** password
5> on
6> go
role
password
Confirm password
1>
```

例 11 アプリケーション名を “isql Session 01” に設定します。

```
isql -UmyAccount -SmyServer --appname "isql Session 01"
Password:
1>select program_name from sysprocesses
2>where spid=@@spid
3>go


program_name
--------------------
isql Session 01
```

例 12 アプリケーション名を isql セッションを開始したスクリプトの名前に設定します。

```
isql --appname $0
```

例 13 この例の *ocs.cfg* ファイルを使用すると、isql を通常どおりに実行したり、ネットワーク・デバッグ情報を使用して実行したりできます。コマンド・ライン・パラメータが読み込まれて解釈された後に、設定ファイルが読み込まれて解釈されるため、CS_APPNAME を *isql* に設定すると、アプリケーション名が isql に戻されます。

```
;Sample ocs.cfg file
[DEFAULT]
;place holder

[isql]
;place holder

[isql_dbg_net]
CS_DEBUG = CS_DBG_NETWORK
CS_APPNAME = "isql"
```

isql を通常どおり実行するには、次のように入力します。

```
isql -Uguest
```

ネットワーク・デバッグ情報を使用して isql を実行するには、次のように入力します。

```
isql -Uguest --appname isql_dbg_net
```

例 14 デフォルト・ログ・ファイルを使用して、コマンド履歴をロードし、保存します。

```
isql -Uguest -Ppassword -Smyase --history p1024
```

例 15 *myaseHistory.log* の内容がメモリにロードされたら、このログ・ファイルを削除します。セッションのコマンド履歴は保管されません。

```
isql -Uguest -Ppassword -Smyase --history 1024
   --history_file myaseHistory.log
```

例 16 コマンド履歴に保管されているすべてのコマンドをリストします。

```
isql -Uguest -Ppassword -Smyase --history p1024
1> h
[1] select @@version
[2] select db_name()
[3] select @@servername
1>
```

例 17 発行済みの最新のコマンドを 2 つリストします。

```
isql -Uguest -Ppassword -Smyase --history p1024
1> h -2
[2] select db_name()
[3] select @@servername
1>
```

例 18 1 とラベル付けされたコマンドをコマンド履歴から再呼び出しします。

```
isql -Uguest -Ppassword -Smyase --history p1024
1> ? 1
1> select @@version
2>
```

例 19 発行済みの最新のコマンドをコマンド履歴から再呼び出しします。

```
isql -Uguest -Ppassword -Smyase --history p1024
1> ? -1
1> select @@servername
2>
```

使用法

- isql プロンプトでは、次のコマンドを使用できます。
  - コマンドを終了する場合：

    ```
    go
    ```

  - クエリ・バッファを消去する場合：

    ```
    reset
    ```

  - オペレーティング・システム・コマンドを実行する場合：

    ```
    !! command
    ```

  - isql を終了する場合：

    ```
    quit
    ```

    または

    ```
    exit
    ```

- SYBASE 環境変数を現在のバージョンの Adaptive Server のロケーションに設定してから、isql を使用してください。
- 5701 サーバ・メッセージ (「データベースが変更されました」) は、ログイン後または use database コマンドの発行後には表示されなくなります。
- エラー・メッセージのフォーマットは、以前のバージョンの isql と異なります。これらのメッセージの値に基づいたルーチンを実行するスクリプトは、書き換えが必要な場合があります。

- isql を対話的に使用するには、オペレーティング・システムのプロンプト画面で isql コマンド (および任意のオプション・フラグ) を入力します。isql プログラムは、SQL コマンドを受け入れ、Adaptive Server に送信します。結果は、フォーマットされ、標準出力に出力されます。isql を終了するには、quit または exit を使用します。
- デフォルトのコマンド・ターミネータ go で始まる行を入力するか、または -c オプションを使用する場合は、他のコマンド・ターミネータで始まる行を入力してコマンドを終了します。コマンド・ターミネータのあとに、コマンドを実行する回数を指定する整数を指定できます。たとえば、select x = 1 を 100 回実行するには、次のように入力します。

  ```
  select x = 1
  go 100
  ```

  結果は、実行の終了時に 1 回表示されます。
- コマンド・ラインにオプションを複数回入力した場合、isql は最後の値を使用します。たとえば、次のコマンドを入力した場合、-c の 2 番目の値 “send” によって最初の値 “.” は無効になります。

  ```
  isql -c. -csend
  ```

  これによって、設定したすべてのエイリアスを無効にすることができます。
- 現在のクエリ・バッファに関してエディタを呼び出すには、行の最初の単語としてエディタ名を入力します。EDITOR 環境変数でエディタを指定して、優先する呼び出し可能なエディタを定義します。EDITOR 環境変数を定義しない場合、デフォルトは edit です。

  たとえば、EDITOR 環境変数を *emacs* に設定している場合は、行の最初の単語として “*emacs*” を使用して、isql からこのエディタを呼び出します。
- オペレーティング・システム・コマンドを実行するには、行頭に “!!” を付けてコマンドを入力します。
- 既存のクエリ・バッファをクリアするには、行に reset とだけ入力します。isql は、保留中の入力内容をすべて破棄します。入力行の任意の場所で [Ctrl + C] を押すことによって、現在のクエリをキャンセルし、isql のプロンプト画面に戻ることもできます。
- isql によって実行されるクエリを含むオペレーティング・システム・ファイルを読み込むには、次のように入力します。

  ```
  isql -Ualma -Ppassword < input_file
  ```

  ファイルには、コマンド・ターミネータが必要です。結果は端末に表示されます。クエリを含むオペレーティング・システム・ファイルを読み込み、結果を別のファイルに書き込むには、次のように入力します。

  ```
  isql -Ualma -Ppassword < input_file > output_file
  ```

- 出力を isql コマンド・ラインからファイルにリダイレクトするには、“>” 演算子と “>>” 演算子を使用します。たとえば、select @@servername コマンドの出力をファイルに書き込むには、次のコマンドを入力します。ファイルが既に存在する場合は、そのファイルが上書きされます。

  ```
  select @@servername
  go > output_file
  ```

  select @@version コマンドの出力を新しいファイルに書き込むには、次のコマンドを入力します。ファイルが既に存在する場合は、出力がそのファイルに追加されます。

  ```
  select @@version
  go >> output_file
  ```

- isql コマンド・ラインで、出力を別のコマンドにパイプするには “|” 演算子を使用します。たとえば、sp_who コマンドの出力を grep にパイプし、文字列 “sa” を含む行を返すには、次のコマンドを入力します。

  ```
  sp_who
  go | grep sa
  ```

- isql のフラグを使用する場合は、大文字と小文字を区別してください。
- isql は float または real データを丸めて、小数点以下 6 桁までを表示します。
- isql を対話的に使用するときは、次のコマンドを使用して、オペレーティング・システム・ファイルをコマンド・バッファに読み込みます。

  ```
  :r filename
  ```

  ファイル内にコマンド・ターミネータを含めないでください。編集が終わったら、ターミネータを対話的に入力します。

- isql を対話的に使用するときは、次のコマンドを使用して、オペレーティング・システム・ファイルをコマンド・バッファに読み込み、表示します。

  ```
  :R filename
  ```

- isql を対話型で使用するときは、次のコマンドを使用して現在のデータベースを変更できます。

  ```
  use databasename
  ```

- isql が Adaptive Server に送信する Transact-SQL 文には、コメントを付けることができます。次の例に示すように、コメントは “/*” と “*/” で囲みます。

```
select au_lname, au_fname
/*retrieve authors' last and first names*/
from authors, titles, titleauthor
where authors.au_id = titleauthor.au_id
and titles.title_id = titleauthor.title_id
/*this is a three-way join that links authors
**to the books they have written.*/
```

  go コマンドをコメント・アウトする場合は、コマンドが行頭にならないようにします。たとえば、go コマンドをコメント・アウトする場合は、次のように入力します。

```
/*
**go
*/
```

  次のように入力しないでください。

```
/*
go
*/
```

- エラー・メッセージのフォーマットは、以前のバージョンの isql と異なります。これらのメッセージの値に基づいたルーチンを実行するスクリプトは、書き換えが必要な場合があります。

表 *A-7: isql* セッション・コマンド

コマンド	説明
reset	クエリ・バッファをクリアする。
quit または exit	isql を終了する。
vi	エディタを呼び出す。
!! *command*	オペレーティング・システム・コマンドを実行する。
:r *filename*	オペレーティング・システム・ファイルを読み込む。
:R *filename*	オペレーティング・システム・ファイルを読み込み、表示する。
use *dbname*	現在のデータベースを *dbname* に変更する。
command > *filename*	実行済みの *command* の出力をファイル *filename* にリダイレクトする。*filename* が既に存在する場合は、ファイルの内容が上書きされる。
command >> *filename*	実行済みの *command* の出力をファイル *filename* にリダイレクトする。*filename* が既に存在する場合は、*command* の出力内容が *filename* に追加される。
command | *application*	Transact-SQL *command* の出力を外部 *application* にパイプする。

コマンド履歴の使用

- コマンド履歴機能は、コマンド・モードでのみ使用できます。また、コマンド履歴に含まれるのは、isql で対話的に発行されたコマンドだけです。コマンド履歴に含まれないコマンドの例として、-i コマンド・ライン・オプションを使用して実行されたコマンドや、次のようにリダイレクトされた入力内容の一部として実行されたコマンドなどがあります。

```
isql -Uguest -Ppassword -Smyase --history p1024
   --history_file myaseHistory.log <<EOF
exec sp_x_y_z
go
EOF
```

- コマンド履歴には、isql セッションで発行された最新のコマンドが含まれます。*history_length* に達すると、isql は最も古いコマンドを履歴から削除し、発行された最も新しいコマンドを追加します。
- 代替ログ・ファイルを指定していない場合や、デフォルト・ログ・ファイルで使用される *$HOME* 環境変数または *%APPDATA%* 環境変数が定義されていない場合は、エラー・メッセージが表示され、コマンド履歴ログは保存されません。

isql セッションで、h [*n*] コマンドを使用してコマンド履歴を表示します。1 ページに最大 24 行のコマンド行を表示できます。コマンド履歴に 25 行以上含まれている場合は、[Enter] キーを押してコマンドの次のセットを表示するか、「a」を入力してすべてのコマンドを 1 ページに表示します。「q」を入力すると、isql に戻ります。

n − 表示するコマンドの数を指定します。*n* が正数の場合は、履歴内の最も古いコマンドから表示されます。*n* が負数の場合は、*n* 個の最新のコマンドが表示されます。

コマンド履歴からコマンドを再呼び出しし、再発行するには、? *n* | ?? コマンドを使用します。

n − *n* が正数の場合、isql は番号 *n* がラベル付けされたコマンドを探し、このコマンドをコマンド・バッファにロードします。*n* が負数の場合、isql は最近発行された *n* 番目のコマンドをロードします。

?? − 発行済みの最新のコマンドを再呼び出しします。これは ? -1 に相当します。

- コマンドを履歴から再呼び出しすると、再呼び出しされたコマンドはコマンド・バッファ内のコマンドを上書きします。
- 再呼び出ししたコマンドを編集してから、サーバに再送信できます。

isql セッションでのプロンプト・ラベルと二重ワイルドカードの使用

isql セッションでは、デフォルトのプロンプト・ラベルはデフォルトのワイルドカード :? または *wildcard* の値のいずれかになります。ワイルドカードの後に 80 文字以内の 1 語の文字列を指定することによって、プロンプト・ラベルをカスタマイズできます。複数語のプロンプト・ラベルを指定した場合、最初の単語の後にある文字は無視されます。

:?:? のような二重ワイルドカードは、isql が同じ入力を要求するプロンプトを 2 回表示する必要があることを指定します。2 回目のプロンプトでは、最初の入力内容を確認するよう求められます。二重ワイルドカードを使用する場合、2 番目のプロンプト・ラベルは “Confirm” で始まります。

注意　isql セッションで、isql が :? または *wildcard* の値をワイルドカードとして認識するのは、これらの文字が isql 行の先頭に配置されている場合だけです。

参照

『ASE リファレンス・マニュアル』の sp_addlanguage、sp_addlogin、sp_configure、sp_defaultlanguage、sp_droplanguage、sp_helplanguage

instjava

説明

クライアント・ファイルから Adaptive Server に JAR ファイルをインストールします。このユーティリティは *%SYBASE%¥%SYBASE_OCS%¥bin* にあります。

構文

```
instjava
   -f file_name
   [ -new | -update ]
   [ -j jar_name ]
   [ -S server_name ]
   [ -U user_name ]
   [ -P password ]
   [ -D database_name ]
   [ -I interfaces_file ]
   [ -a display_charset ]
   [ -J client_charset ]
   [ -z language ]
   [ -t timeout ]
   [-v]
```

パラメータ

-f *file_name*

データベースにインストールするクラスが入っているソース・ファイルの名前を指定します。

-new | -update

ソース・ファイル内のクラスがデータベースにすでに存在するかどうかを指定します。

new パラメータを指定する場合は、既存のクラスと同じ名前のクラスをインストールすることはできません。

update パラメータを指定すると、既存のクラスと同じ名前のクラスをインストールし、既存のクラスを新しくインストールしたクラスに置き換えることができます。

-j *jar_name*

データベースにインストールするクラスが含まれている JAR ファイルの名前を指定します。このパラメータを指定すると、JAR ファイルがデータベースに保存され、データベースに格納されているクラスに関連付けられます。

-S *server_name*

サーバの名前を指定します。

-U *user_name*

Adaptive Server ログイン名を指定します。-U フラグとパラメータを省略した場合、またはパラメータを指定せずに -U フラグを指定した場合は、Adaptive Server は現在のユーザのオペレーティング・システムのログイン名を使用します。

-P *password*

Adaptive Server のパスワードを指定します。-P フラグとパラメータを省略した場合、instjava はパスワードの入力を求めるプロンプトを表示します。パスワードなしで -P フラグを指定すると、null パスワードが使用されます。

-D *database_name*

JAR ファイルをインストールするデータベースの名前を指定します。-D フラグを省略した場合、またはパラメータを指定せずに -D フラグを指定した場合は、ユーザのデフォルト・データベースが使用されます。

-I *interfaces_file*

Adaptive Server に接続するときに検索する interfaces ファイルの名前とロケーションを指定します。-I フラグとパラメータを省略した場合、またはパラメータを指定せずに -I フラグを指定した場合は、SYBASE 環境変数で指定されたディレクトリにある interfaces ファイルが使用されます。

-a *display_charset*

サーバの文字セットと異なる文字セットを使用するマシンから、instjava を使用できるようにします。-a を -J とともに使用して、変換に必要な文字セット変換ファイル (*.xlt* ファイル) を指定します。-J を指定しないで -a を使用するのは、クライアントの文字セットがデフォルトの文字セットと同じである場合だけです。

-J *client_charset*

クライアントで使用する文字セットを指定します。instjava は、フィルタを使用して *client_charset* と Adaptive Server の文字セット間で入力を変換します。

-J *client_charset* は、クライアントで使用する文字セットである *client_charset* とサーバの文字セット間の変換を Adaptive Server に要求します。

-J に引数を指定しないと、文字セット変換が無効になります。クライアントとサーバが同じ文字セットを使用する場合に、このパラメータを使用してください。

-J を省略すると、文字セットはプラットフォームのデフォルトに設定されます。デフォルトの文字セットは、クライアントが使用している文字セットと同じであるとはかぎりません。文字セットおよび関連するフラグの詳細については、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。

-z *language*

instjava のプロンプトとメッセージの表示に使用する代替言語の名前を指定します。-z フラグを指定しない場合、instjava はサーバのデフォルト言語を使用します。言語はインストール時に Adaptive Server に追加できます。インストール後でも、langinstall ユーティリティまたは sp_addlanguage ストアド・プロシージャを使用して追加できます。

-t *timeout*

SQL コマンドがタイムアウトするまでの秒数を指定します。タイムアウトを指定しないと、コマンドは無期限に実行されます。このパラメータは、接続時間ではなく、instjava 内から発行されたコマンドに影響します。instjava にログインするためのデフォルトのタイムアウトは 60 秒です。

-v

instjava のバージョン番号と版権メッセージを表示して、終了します。

例

例 1 *addr.jar* とそのクラスをインストールしますが、JAR とクラスの関連付けは保持しません。

```
instjava -f '¥home¥usera¥jars¥addr.jar' -new
```

例 2 *addr.jar* を再インストールし、そのクラスを employees という JAR 名に関連付けます。

```
instjava -f '¥home¥usera¥jars¥addr.jar' -update -j employees
```

使用法

- SYBASE 環境変数を現在のバージョンの Adaptive Server のロケーションに設定してから、instjava を使用してください。
- データベースで Java が有効になっている場合の instjava の使用方法については、『Adaptive Server Enterprise における Java』を参照してください。
- インストール済みのクラスは、すべてのユーザが参照できます。
- パラメータ・フラグ -f、-j、-S、-U、-P、-D、-I を指定する場合、フラグとその後のパラメータの間にはスペースを入れても入れなくてもかまいません。

新規 JAR の追加

- -jar オプションを指定して -new を使用したときに、その名前の JAR ファイルがデータベースに既に存在する場合は、例外が発生します。
- ソース JAR 内のクラスと同じ名前のクラスがデータベースに既に存在する場合は、例外が発生します。

JAR とクラスの更新

警告！ 修正したバージョンのクラスを再インストールして、カラムのデータ型として使用しているクラスを変更した場合、そのクラスをデータ型として使用しているテーブルの既存のオブジェクト (ロー) を、変更したクラスが読み込んだり使用したりできることを確認してください。できなかった場合は、クラスを再インストールしないと現在のオブジェクトにアクセスできません。

- -jar オプションを指定して -update を使用すると、次のようになります。
  - データベース内のクラスの中で、ターゲット JAR に関連付けられたすべてのクラスがデータベースから削除され、代わりにソース JAR ファイル内のクラスがインストールされます。
  - ソース JAR ファイル内に、データベースにインストールされていても JAR には付加されていないクラスがある場合、ソース JAR ファイル内のクラスがデータベースにインストールされ、付加されていないクラスは削除されます。

- -jar オプションを指定せずに、-update を使用すると、次のようになります。
  - ソース JAR ファイル内のクラスが、同じ名前の付加されていないクラスと置き換わります。
  - ソース JAR ファイル内のインストールされていないクラスは、付加されていないクラスとしてデータベースにインストールされます。
- SQLJ プロシージャまたは関数が参照しているインストール済みのクラスに代わる新しい JAR をインストールする場合は、新しくインストールされたクラスに SQLJ ルーチンの有効なシグニチャがあることを確認してください。シグニチャが無効の場合、SQLJ ルーチンが呼び出されると例外が発生します。

ロック

- instjava を実行すると、sysxtypes に排他ロックが設定されます。
- -jar を指定すると、sysjars に排他テーブル・ロックが設定されます。

パーミッション　instjava を使用できるのは、システム管理者とデータベース所有者だけです。

Tables used　sysjars、sysxtypes

参照

システム・プロシージャ　sp_helpjava

ユーティリティ　extrjava

extrjava

説明

保持されている JAR ファイルと、ファイルに含まれるクラスを Adaptive Server からクライアント・ファイルにコピーします。このユーティリティは *%SYBASE%¥%SYBASE_OCS%¥bin* にあります。

構文

```
extrjava
   -j jar_name
   -f file_name
   [-S server_name]
   [-U user_name]
   [-P password]
   [-D database_name]
   [-I interfaces_file]
   [-a display_charset]
   [-J client_charset]
   [-z language]
   [-t timeout]
   [-v]
```

パラメータ

-j *jar_name*

転送元データベースに保持されている JAR に割り当てられた名前を指定します。

-f *file_name*

転送先のクライアント・ファイルの名前を指定します。

-S *server_name*

サーバの名前を指定します。

-U *user_name*

Adaptive Server ログイン名を指定します。-U フラグとパラメータを省略した場合、またはパラメータを指定せずに -U フラグを指定した場合は、Adaptive Server は現在のユーザのオペレーティング・システムのログイン名を使用します。

-P *password*

Adaptive Server のパスワードを指定します。-P フラグとパラメータを省略した場合、extrjava はパスワードの入力を求めるプロンプトを表示します。パスワードなしで -P フラグを指定すると、null パスワードが使用されます。

-D *database_name*

JAR をインストールするデータベースの名前を指定します。-D フラグを省略した場合、またはパラメータを指定せずに -D フラグを指定した場合は、ユーザのデフォルト・データベースが使用されます。

-I *interfaces_file*

Adaptive Server に接続するときに検索する interfaces ファイルの名前とロケーションを指定します。-I フラグとパラメータを省略した場合、またはパラメータを指定せずに -I フラグを指定した場合は、SYBASE 環境変数で指定されたディレクトリにある interfaces ファイルが使用されます。

-a *display_charset*

サーバの文字セットと異なる文字セットを使用するマシンから、extrjava を使用できるようにします。-a を -J とともに使用して、変換に必要な文字セット変換ファイル (*.xlt* ファイル) を指定します。-J を指定しないで -a を使用するのは、クライアントの文字セットがデフォルトの文字セットと同じである場合だけです。

-J *client_charset*

クライアントで使用する文字セットを指定します。extrjava は、フィルタを使用して *client_charset* と Adaptive Server の文字セット間で入力を変換します。

-J *client_charset* は、クライアントで使用する文字セットである *client_charset* とサーバの文字セット間の変換を Adaptive Server に要求します。

-J に引数を指定しないと、文字セット変換が無効になります。クライアントとサーバが同じ文字セットを使用する場合に、このパラメータを使用してください。

-J を省略すると、文字セットはプラットフォームのデフォルトに設定されます。デフォルトの文字セットは、クライアントが使用している文字セットと同じであるとはかぎりません。文字セットおよび関連するフラグの詳細については、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。

-z *language*

extrjava のプロンプトとメッセージの表示に使用する代替言語の名前を指定します。-z フラグを指定しない場合、extrjava はサーバのデフォルト言語を使用します。言語はインストール時に Adaptive Server に追加できます。インストール後でも、langinstall ユーティリティまたは sp_addlanguage ストアド・プロシージャを使用して追加できます。

-t *timeout*

SQL コマンドがタイムアウトするまでの秒数を指定します。タイムアウトを指定しないと、コマンドは無期限に実行されます。このパラメータは、接続時間ではなく、extrjava 内から発行されたコマンドに影響します。extrjava にログインするためのデフォルトのタイムアウトは 60 秒です。

-v

extrjava のバージョン番号と版権メッセージを表示して、終了します。

例

employees という名前の JAR に関連付けられたクラスを、クライアント・ファイル *newaddr.jar* にダウンロードします。

```
extrjava -j employees -f '¥home¥usera¥jars¥addr.jar' -new
```

使用法

- SYBASE 環境変数を現在のバージョンの Adaptive Server のロケーションに設定してから、extrjava を使用してください。
- ターゲット・クライアント・ファイルがすでに存在する場合、extrjava によってファイルの内容は上書きされます。
- パラメータ・フラグ -f、-j、-S、-U、-P、-D、-I を指定する場合、フラグとその後のパラメータの間にはスペースを入れても入れなくてもかまいません。
- extrjava を実行すると、sysxtypes に排他ロックが設定されます。
- -jar を指定すると、sysjars に排他テーブル・ロックが設定されます。
- データベースで Java が有効になっている場合の extrjava の使用方法については、『Adaptive Server Enterprise における Java』を参照してください。

パーミッション

extrjava を使用できるのは、システム管理者とデータベース所有者だけです。

Tables used

sysjars、sysxtypes

参照

システム・プロシージャ sp_helpjava

ユーティリティ instjava

索引

E

F

G

H

I

L

M

O

P

R

S

T

U

W

い

か

こ

さ

し

す

せ

て

と

は

ふ

へ

ま

も

ゆ

ら