SYBASE®
An SAP® Company

プログラマーズ・ガイド補足

Open Client™/Open Server™

15.7

[UNIX 版]

ドキュメント ID ： DC35453-01-1570-01
改訂： 2012 年 6 月

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

目次

はじめに

Sybase® Open Client™ および Open Server™ 製品は、アプリケーションと任意のデータ型をともに使用できる、プログラミング・インタフェースのセットです。次の製品が用意されています。

- Open Client DB-Library™/C
- Open Client Client-Library/C
- Open Server Server-Library/C
- Open Client Embedded SQL™/C
- Open Client Embedded SQL/COBOL

これらの各製品には、製品の詳細を説明する独自のリファレンス・マニュアルがあります。このマニュアルの目的は、製品マニュアルを補足することです。このマニュアルは、すべての Open Client/Server 製品について、プラットフォームに関連した問題を説明します。

対象読者

このマニュアルは、上記の Open Client/Server 製品を使用するプログラマの方を対象としています。

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 Client-Library/C」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 Open Client DB-Library/C」では、DB-Library のサンプル・プログラム、および実行プログラムを構築する方法について説明します。
- 「第 3 章 Open Server Server-Library/C」では、Server-Library のサンプル・プログラム、および実行プログラムを構築する方法について説明します。
- 「第 4 章 Open Client Embedded SQL/C」では、Embedded SQL/C のサンプル・プログラム、および実行プログラムを構築する方法について説明します。

- 「第 5 章 Open Client Embedded SQL/COBOL」では、Embedded SQL/COBOL のサンプル・プログラム、および実行プログラムを構築する方法について説明します。
- 「付録 A ユーティリティ・コマンド・リファレンス」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・ページで構成されています。
- 「付録 B 環境変数」では、アプリケーションを構築し、実行するために設定する必要がある環境変数について説明します。
- 「付録 C ユーティリティ・メッセージ」では、bcp、defncopy、isql の各ユーティリティのエラー、情報、警告のメッセージについての情報を提供します。

関連マニュアル

詳細については、これらのマニュアルを参照できます。

- 『Open Server および SDK 新機能』(各 Windows、Linux、UNIX 版) では、Open Server と Software Developer's Kit の新機能について説明しています。このマニュアルは、新機能の提供に伴って改訂されます。
- 使用しているプラットフォームの Open Server の『リリース・ノート』には、Open Server に関する重要な最新情報が記載されています。
- 使用しているプラットフォームの『Software Developer's Kit リリース・ノート』には、Open Client™ および SDK に関する重要な最新情報が記載されています。
- 『jConnect™ for JDBC™ リリース・ノート』には、jConnect に関する重要な最新情報が記載されています。
- 使用しているプラットフォームの『Open Client/Server 設定ガイド』では、システムを設定して Open Client/Server 製品を実行する方法について説明しています。
- 『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 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 Server Server-Library/C リファレンス・マニュアル』では、Open Server Server-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』では、CS-Library のリファレンス情報について説明しています。CS-Library は、Client-Library と Server-Library の両方のアプリケーションで役に立つユーティリティ・ルーチンの集まりです。
- 『Open Server DB-Library/C リファレンス・マニュアル』では、C バージョンの Open Client DB-Library™ のリファレンス情報について説明しています。
- 『Sybase® SDK DB-Library Kerberos 認証オプションのインストールおよびリリース・ノート』では、DB-Library で使用する MIT Kerberos セキュリティ メカニズムをインストールして有効化にする方法について説明しています。DB-Library でサポートされる Kerberos セキュリティ・メカニズムの機能は、ネットワーク認証サービスと相互認証サービスのみです。
- 『Open Client/Server 開発者用国際化ガイド』では、国際化されたアプリケーションとローカライズされたアプリケーションを作成する方法について説明しています。
- 『Open Client Embedded SQL™/C プログラマーズ・ガイド』では、C アプリケーションで Embedded SQL および Embedded SQL プリコンパイラを使用する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、COBOL アプリケーションで Embedded SQL および Embedded SQL プリコンパイラを使用する方法について説明しています。
- 『jConnect for JDBC プログラマーズ・リファレンス』では、jConnect for JDBC 製品について説明し、リレーショナル・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する方法について説明しています。
- Sybase® 製 *Adaptive Server® Enterprise ODBC* ドライバの『ユーザーズ・ガイド』(Microsoft Windows および UNIX 版) では、Microsoft Windows および UNIX プラットフォームの Adaptive Server から、Open Database Connectivity (ODBC) ドライバを使用してデータにアクセスする方法について説明します。
- 『Perl 用 Adaptive Server Enterprise データベース・ドライバ・プログラマーズ・ガイド』では、Perl 開発者が Perl スクリプトを使用して Adaptive Server のデータベースに接続し、情報をクエリまたは変更する方法について説明しています。

- 『PHP 用 Adaptive Server Enterprise 拡張モジュール・プログラマーズ・ガイド』では、PHP 開発者が Adaptive Server データベースに対してクエリを実行する方法について説明しています。
- 『Python 用 Adaptive Server Enterprise 拡張モジュール・プログラマーズ・ガイド』では、Adaptive Server データベースに対してクエリを実行するときに使用できる Sybase 固有の Python インタフェースについて説明しています。

その他の情報

Sybase Getting Started CD および Sybase Product Documentation 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、リリース・ノートとインストール・ガイドが PDF 形式で含まれています。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)。
- Sybase Product Documentation Web サイトには、標準の Web ブラウザを使用してアクセスできます。また、製品ドキュメントのほか、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 Documentation Web サイトは、Product Documentation (http://www.sybase.com/support/manuals/) にあります。

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 (http://www.sybase.com/support/techdocs/) を指定します。
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 (http://certification.sybase.com/) を指定します。

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 (http://www.sybase.com/support/techdocs/) を指定します。

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 the Sybase Support Page (http://www.sybase.com/support) を指定します。

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 に、このマニュアルで使用されている構文の表記規則を示します。

表 1：構文の表記規則

凡例	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。
( )	この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 Client-Library/C

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でき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。

Open Client 製品に関する追加情報および使用するプラットフォームでの動作については、最新リリースの Software Developer's Kit の『リリース・ノート』を参照してください。

Open Client Client-Library/C を使用できるオペレーティング・システム・プラットフォームのリストについては、『Open Server および SDK 新機能』(各 Windows、Linux、UNIX 版) を参照してください。

一般的な条件

Client-Library のサンプル・プログラムを実行するには、以下の準備が必要です。

- Adaptive Server® Enterprise に接続できる必要があります。『Open Client/Server 設定ガイド UNIX 版』を参照してください。また、必要な Adaptive Server のバージョン・レベルについては、それぞれのサンプル・プログラムの説明を参照してください。
- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。
  - SYBASE
  - SYBASE_OCS
  - DSQUERY
  - SYBPLATFORM
  - プラットフォーム固有のライブラリ・パス変数
- サンプル・プログラムを実行する方法の詳細については、*$SYBASE/$SYBASE_OCS/sample/ctlibrary* ディレクトリにある *README* ファイルを参照してください。

Client-Library 実行プログラムの構築

マルチスレッド・アプリケーションをはじめとする Client-Library アプリケーションを構築するには、ライブラリ、およびコンパイルとリンク行を使用します。

表 1-1 に、非スレッド環境ですべての Client-Library 機能を十分に活用するために組み込む必要のあるライブラリを示します。

表 1-1: 非スレッド環境のライブラリ

プラットフォーム	必要なライブラリ
すべてのプラットフォーム	*libsybct* ― Client-Library (Sybase) *libsybcs* ― CS-Library (Sybase) *libsybtcl* ― トランスポート制御層 (Sybase 内部使用) *libsybcomn* ― 内部共有ユーティリティ・ライブラリ (Sybase 内部使用) *libsybintl* ― 国際化サポート・ライブラリ (Sybase 内部使用) *libsybunic* ― Unicode-Library (Sybase 内部使用)

ネイティブ・スレッドのサポート

Client-Library には、スレッドセーフ・ライブラリが含まれています。開発者は、これらのライブラリを使用して POSIX スレッドを使用するマルチスレッド・アプリケーションを作成できます。

適切な構文と例については、「マルチスレッド・アプリケーションのコンパイルとリンク行」(9 ページ) を参照してください。

表 1-2 に、マルチスレッド・サポート用のすべての Client-Library 機能を利用するために組み込む必要があるライブラリを示します。

表 1-2: マルチスレッド・サポート用のプラットフォーム固有ライブラリ

プラットフォーム	必要なライブラリ
すべてのプラットフォーム	*libsybct_r* ― Client-Library (Sybase) *libsybcs_r* ― CS-Library (Sybase) *libsybintl_r* ― 国際化サポート・ライブラリ (Sybase 内部使用) *libsybtcl_r* ― トランスポート制御層 (Sybase 内部使用) *libsybcomn_r* ― 内部共有ユーティリティ・ライブラリ (Sybase 内部使用)
Solaris プラットフォーム	*libthread* ― ネイティブ・スレッド・ライブラリ (システム) *libpthread* ― スレッド・ライブラリ (システム) *libsocket* ― ソケット・ネットワーク・ライブラリ (システム) *libnsl* ― ネットワーク・ライブラリ (システム) *libdl* ― 動的ローダ・ライブラリ (システム)
HP HP-UX プラットフォーム	*libcl* ― HP トランスポート制御層 (システム) *libBSD* ― BSD ライブラリ (システム) *libc_r* ― C 言語リエントラント・ライブラリ *libdld* ― 動的ローダ・ライブラリ (システム)

プラットフォーム	必要なライブラリ
IBM AIX プラットフォーム	*libc_r* － C 言語リエントラント・ライブラリ *libpthreads* － スレッド・ライブラリ (システム)
Linux プラットフォーム	*libpthread* － スレッド・ライブラリ (システム)

Kerberos サポート

Client-Library は、ネットワークを介して通信するときに高度なセキュリティを必要とするアプリケーションに対して Kerberos セキュリティ機能をサポートします。必要な Kerberos ソフトウェアをインストールし、適切な設定作業を実行することによって、Client-Library アプリケーションは次の Kerberos セキュリティ機能を利用できます。

- ネットワーク認証
- 相互認証
- 順序不整合認証
- リプレイの検出
- 機密性
- 整合性
- クレデンシャルの委任

表 1-3: Kerberos サポートに必要な作業

タスク	参照先
Kerberos ソフトウェアをシステムにインストールする。	Kerberos のマニュアルおよび『Open Client/Server 設定ガイド UNIX 版』を参照。
libtcl.cfg 設定ファイルのセキュリテイ・セクションを設定します。	『Open Client/Server 設定ガイド UNIX 版』を参照してください。
Kerberos ユーティリティ kinit を使用して Kerberos セキュリティ環境にログインし、Client-Library アプリケーションを実行する。	Kerberos のマニュアルを参照。

タスク	参照先
環境変数にクレデンシャル・キャッシュ・ディレクトリのロケーションを設定する。 • CyberSafe の場合、CSFC5CCNAME • MIT の場合、KRB5CCNAME	Kerberos のマニュアルを参照。クレデンシャル・キャッシュ・ディレクトリのデフォルト・ロケーションはプラットフォームごとに異なる。
ct_con_props を使用して必要なセキュリティ機能を設定する、または ct_con_props を設定しないでデフォルト・クレデンシャルを使用する。	『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。 セキュリティ機能がサポートされているかどうかを調べるには、ct_con_props と ct_config 内の CS_SUPPORTED アクション・タイプを使用する。

コンパイルとリンク行

Client-Library と Server-Library は、ディレクトリ・ドライバとセキュリティ・ドライバを動的にリンクします。アプリケーションに Sybase のディレクトリ・ドライバまたはセキュリティ・ドライバ (リンカ・オプションは -lsybdldap と -lsybskrb) を明示的にリンクしないでください。

非スレッド・アプリケーション用のコンパイルとリンク行

以下の表に、UNIX 上で稼働し、Sybase がサポートするプラットフォーム上で、非スレッド Client-Library アプリケーションのコンパイルとリンクを行うためのコマンドの一般的なフォーマットを示します。コンパイルとリンクの詳細については、*$SYBASE/$SYBASE_OCS/sample/ctlibrary* 内の *makefile* および *sybopts.sh* ファイルを参照してください。

表 1-4 に、静的ライブラリを使用して Client-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 1-4: Client-Library の静的なコンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro6.2/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,archive -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,default -lcl -lm -lBSD -ldld -Wl,-E,+s -o program`
HP HP-UX Itanium 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,archive -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,default -lcl -lm -lBSD -ldld -Wl,-E,+s -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`

プラットフォーム	コマンド
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc -I$SYBASE/$SYBASE_OCS/include L$SYBASE/$SYBASE_OCS/lib program.c -lsybct64 -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -lld -lnsl -lm64 -o program`

表 1-5 に、デバッグ・ライブラリを使用して Client-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 1-5: Client-Library のデバッグ・コンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -ldld -o program`

プラットフォーム	コマンド
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -ldld -o program`
Linux x86 32 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct64 -l sybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -lld -lnsl -lm64 -o program`

表 1-6 に、共有ライブラリを使用して Client-Library アプリケーションのコンパイルとリンク (動的ドライバを使用) を行うためのコマンドを示します。

表 1-6: Client-Library の共有コンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/$SYBASE_OCS/lib program.c -Bdynamic -lsybct -lsybcs -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/$SYBASE_OCS/lib program.c -Bdynamic -lsybct -lsybcs -lnsl -ldl -lm -lsocket -o program`

プラットフォーム	コマンド
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct -lsybcs -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybct -lsybcs -lcl -lm -lBSD -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybct -lsybcs -lcl -lm -lBSD -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct64 -lsybcs64 -ldl -lnsl -lm64 -o program`

マルチスレッド・アプリケーションのコンパイルとリンク行

表 1-7 に、スレッドセーフ・サポートを利用するために、Client-Library アプリケーションをコンパイルしてライブラリにリンクするためのコマンドを示します。

表 1-7: Client-Library のスレッドセーフのコンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_REENTRANT program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lnsl -ldl -lpthread -lthread -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_REENTRANT program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lnsl -ldl -lpthread -lthread -lm -lsocket -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc_r -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lpthread -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -D_REENTRANT -Ae program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lcl -lm -lBSD -lpthread -ldld -o program`
HP HP-UX Itanium 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -D_REENTRANT -Ae program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lcl -lm -lBSD -lpthread -ldld -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -ldl -lpthread -lnsl -lm -o program`

プラットフォーム	コマンド
Linux POWER 32 ビット版および 64 ビット版	`xlc_r -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -ldl -lpthread -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc_r -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct64_r -lsybcs64_r -lsybtcl64_r -lsybcomn64_r -lsybintl64_r -ldl -lpthread -lnsl -lm64 -o program`

HP HP-UX システムのユーザの場合

- -Wl,-a,archive オプションを使用すると、リンカは Sybase ライブラリを静的にリンクします。このオプションを指定していない場合、Client-Library は Sybase ライブラリの共有バージョンを使用します。共有ライブラリを使用する場合は、実行時に SHLIB_PATH 環境変数に *$SYBASE/$SYBASE_OCS/lib* を設定する必要があります。また、アプリケーション・ユーザには、*$SYBASE/$SYBASE_OCS/lib* 内のライブラリに対する read と execute のパーミッションが必要です。
- アプリケーションが +s リンカ・オプションを使用してリンクされている場合を除き、HP HP-UX は実行時に SHLIB_PATH 環境変数を使用しません。システムが実行時に Sybase ライブラリを見つけることができるようにするには、+s リンカ・オプションを使用してください。-E は、実行時にドライバ・ライブラリがロードされるときに、未定義シンボル・エラーにならないようにするために必要です。HP-UX ld の man ページを参照してください。
- 環境変数 LD_LIBRARY_PATH に *$SYBASE/$SYBASE_OCS/lib* を設定して、共有 (動的) ライブラリにリンクされたプログラムを実行してください。デバッグ・モードでプログラムを実行する場合は、LD_LIBRARY_PATH に *$SYBASE/$SYBASE_OCS/devlib* を設定します。

  LD_LIBRARY_PATH はプラットフォーム固有です。表 1-8 に、各プラットフォームの環境変数を示します。

表 1-8: 各 UNIX プラットフォームの LD_LIBRARY_PATH

プラットフォーム	環境変数
Solaris SPARC 32 ビット版、Solaris x86-64 32 ビット版、HP HP-UX PA-RISC 64 ビット版、HP HP-UX Itanium 64 ビット版、Linux x86 32 ビット版、Linux POWER 32 ビット版および 64 ビット版、Linux x86-64 64 ビット版	LD_LIBRARY_PATH
Solaris SPARC 64 ビット版、Solaris x86-64 64 ビット版	LD_LIBRARY_PATH_64
HP HP-UX PA-RISC 32 ビット版、HP HP-UX Itanium 32 ビット版	SHLIB_PATH
IBM AIX RS/6000 32 ビット版および 64 ビット版	LIBPATH

Kerberos サポート・アプリケーションのコンパイルとリンク行

Kerberos 用の Sybase ドライバは、動的にロードされる共有ライブラリです。ドライバがロードされると、Kerberos GSS ライブラリが動的にロードされます。ライブラリは、動的ローダが使用する検索パスに存在する必要があります。Sybase ドライバの実装に関する制約により、Kerberos を使用しているときはリエントラント・ライブラリだけがサポートされます。

バルク・コピー・ルーチン

バルク・コピー・ルーチンを使用する場合は、*libsybblk* ライブラリをリンクします。スレッド・アプリケーションでバルク・コピー・ルーチンを使用する場合は、*libsybblk_r* ライブラリをリンクします。

バルク・コピー・ライブラリをリンクするには次のようにします。

- 非スレッド・アプリケーションでは、リンク行の -lsybct の前に -lsybblk を追加します。
- マルチスレッド・アプリケーションでは、リンク行の -lsybct_r の前に -lsybblk_r を追加します。

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

パフォーマンスの考慮事項

共有ライブラリとリンクすると、静的ライブラリとリンクする場合よりも実行プログラムが小さくなり、リンク時間も少なくて済みます。ただし、共有ライブラリとリンクされた実行プログラムは、静的ライブラリを使用してリンクされた実行プログラムよりも起動に時間がかかります。さらに、静的ライブラリとは違って共有ライブラリは実行時に使用可能でなければなりません。

最高のパフォーマンスを提供するライブラリのタイプは、個々のサイトの稼働条件によって決まります。

ヘッダ・ファイル

ctpublic.h ヘッダ・ファイルは、すべての Client-Library アプリケーションのソース・ファイルにインクルードする必要があります。他の必要なヘッダ・ファイルは、*ctpublic.h* 内にネストされています。Bulk-Library を使用する場合は、*ctpublic.h* ではなく *bkpublic.h* をインクルードしてください。

『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。

Client-Library のサンプル・プログラムの使用

Client-Library には、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 で提供されるサンプル・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、使用するプラットフォームの『Adaptive Server Enterprise 15.7 インストール・ガイド』を参照してください。必要なデータベースについては、各サンプル・プログラムの前提条件の説明を参照してください。

makefile とサンプル・プログラム

makefile を使用して、すべてのプラットフォームでサンプル・プログラムを構築するには、使用しているコンパイラに合わせて SYBPLATFORM 環境変数を正しく設定してください。詳細については、表 B-1 (172 ページ) を参照してください。

サンプル・プログラムの目的

サンプル・プログラムは、Client-Library 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 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。実際の運用環境で使用できるレベルのプログラムでは、エラーや特殊なケースを処理するためのコードを追加する必要があります。

---

sybopts.sh スクリプトとアプリケーションの構築

sybopts.sh は、Open Client/Open Server アプリケーションの構築に役立つ SYBPLATFORM 環境変数を読み込みます。

```
sybopts.sh args
```

args には次のいずれかの引数を指定します。

- compile ― コンパイラのコマンドとプラットフォーム固有のコンパイル・フラグを返す。
- comlibs ― アプリケーションにリンクさせなければならない必須の Sybase ライブラリのリストを返す。
- syslibs ― アプリケーションにリンクさせなければならない Sybase 以外の必須ライブラリのリストを返す。

すべての引数 (*args*) のリストについては、*$SYBASE/$SYBASE_OCS/sample/ctlibrary* にある *sybopts.sh* スクリプトの "Usage" セクションを参照してください。

ロケーション

サンプル・プログラムは、*$SYBASE/$SYBASE_OCS/sample/ctlibrary* にあります。

このディレクトリには次のファイルが含まれています。

- サンプル・プログラムのソース・コード
- サンプル・プログラム用のデータ・ファイル
- サンプル・プログラムを構築するための *makefile*。*makefile* は、Client-Library アプリケーションの作成を開始するときに使用します。
- サンプル・プログラムのヘッダ・ファイル − *example.h*、*ctxact.h*、*exasync.h*、*exutils.h*、*thrdfuc.h*、*thrdutil.h*、*wide_example.h*
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

$SYBASE/$SYBASE_OCS/sample/ctlibrary の内容を、元のファイルの整合性に影響を与えないでサンプル・プログラムを自由に使用できるような作業ディレクトリにコピーしてから、コンパイルして実行してください。

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 *example.h* を参照します。このファイル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 sample 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

/*
** 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         ""
```

サンプル・プログラムは、次のコード例に示すように *example.h* 内の define 文を使用します。

```
CS_CHAR *Ex_username = EX_USERNAME;
CS_CHAR *Ex_password = EX_PASSWORD;

/*
** If a user name is defined, set the
** CS_USERNAME property.
*/
if (retcode == CS_SUCCEED && Ex_username != NULL)
{
    if ((retcode = ct_con_props(*connection,
        CS_SET, CS_USERNAME, Ex_username,
        CS_NULLTERM, NULL)) != CS_SUCCEED)
    {
        ex_error("ct_con_props(username) failed");
    }
}

/*
** If a password is defined, set the
** CS_PASSWORD property.
*/
if (retcode == CS_SUCCEED && Ex_password != NULL)
{
    if ((retcode = ct_con_props(*connection,
        CS_SET, CS_PASSWORD, Ex_password,
        CS_NULLTERM, NULL)) != CS_SUCCEED)
```

```
        {
            ex_error("ct_con_props(password) failed");
        }
    }
```

EX_USERNAME は、*example.h* 内で "sa" と定義されています。サンプル・プログラムを実行する前に、*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* 内で null (" ") 文字列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、*example.h* を編集して null (" ") 文字列を
サーバのパスワードに変更することができます。

EX_PASSWORD に関しては、次の 3 つのオプションがあります。必要に応じて適切なものを選択してください。

- 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 (" ") 文字列に変更します。このオプションは、セキュリティを侵害される可能性があります。パスワードがこのような公開された値に設定されていると、承認されていないユーザでもサーバにログインできるからです。これでは問題がある場合は、次の 2 つの方法のどちらかを選択してください。
- *example.h* 内で、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 コードを修正して、サンプル・プログラムのユーザにサーバのパスワードの入力を求めるプロンプトを表示するコードで置き換えます。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・プログラムのためのユーティリティ・ルーチン

exutils.c ファイルには、Client-Library の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。

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。

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 ルーチンは、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 ルーチンは、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 ルーチンは、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 ルーチンは、戻される結果タイプを処理します。
- fetch_n_print ルーチンは、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。

サンプル・プログラムの概要

特に指定がない場合、各サンプル・プログラムの追加情報については、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。

arraybind.c サンプル・プログラム

arraybind.c サンプル・プログラムは、ct_command によって起動した 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* プログラムは標準の 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。

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。

---

batch_lang.c サンプル・プログラム

batch_lang.c サンプル・プログラムは、言語文で ct_send_params() を使用する方法を示します。このサンプル・プログラムでは、ct_send_params() を繰り返し使用して、ファイルから読み込まれた行をテーブルに挿入します。読み込まれるすべての行のパラメータで同じ位置が使用されるため、ct_send_params() の呼び出しの間に ct_param() または ct_setparam() を呼び出す必要はありません。

batch_dynamic.c サンプル・プログラム

batch_dynamic.c サンプル・プログラムでは、動的 SQL を使用して、異なるメモリ位置にデータが存在するサーバにパラメータを送信します。このため、このサンプル・プログラムは、ct_setparam() を使用して異なる変数に再バインドしてから ct_send_params() を再び呼び出す方法も示しています。

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。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。

compute.c サンプル・プログラム

compute.c サンプル・プログラムは、計算結果を処理する方法を示します。このプログラムは次のように動作します。

- 言語コマンドを使用してクエリを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。

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。

サーバに送信される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)
  ```

csr_disp.c サンプル・プログラム

csr_disp.c サンプル・プログラムは、読み込み専用カーソルの使い方を示します。このプログラムは次のように動作します。

- 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。

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。

クエリは次のとおりです。

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

csr_disp_scrollcurs.c サンプル・プログラム

csr_disp_scrollcurs.c サンプル・プログラムは、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。このプログラムは次のように動作します。

- 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。

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。

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
```

csr_disp_scrollcurs2.c サンプル・プログラム

csr_disp_scrollcurs2.c サンプル・プログラムは、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。このプログラムは次のように動作します。

- 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。

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。

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
```

csr_disp_implicit.c サンプル・プログラム

csr_disp_implicit.c サンプル・プログラムは、暗黙的読み込み専用カーソルの使い方を示します。このプログラムは次のように動作します。

- 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。

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。

このサンプル・プログラムの動作は、*csr_disp.c* サンプル・プログラムと同じです。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
```

ex_alib.c と *ex_amain.c* サンプル・プログラム

このサンプル・プログラムは、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完了コールバックの使用を可能にします。

このサンプル・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* は、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。ex_alib.*c* は、1 回の非同期オペレーションでクエリをサーバに送信し、サーバから結果を取得します。
- *ex_amain.c* には、*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。

サンプル・ソース・ファイルと *EX_AREAD.ME* ファイルの先頭にあるコメントを参照してください。

exconfig.c サンプル・プログラム

exconfig.c サンプル・プログラムは、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、デフォルト・ランタイム設定ファイル (*$SYBASE/$SYBASE_OCS/config/ocs.cfg*) を編集する必要があります。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 *$SYBASE/$SYBASE_OCS/config/ocs.cfg* を作成して、次のセクションを追加します。

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

real_server_name には、接続先の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。

firstapp.c サンプル・プログラム

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』を参照してください。

getsend.c サンプル・プログラム

getsend.c サンプル・プログラムは、さまざまなデータ型を含むテーブルから text データを取り出して更新する方法を示します。ここに示すのと同じプロセスを使用して、image データを取り出して更新できます。

i18n.c サンプル・プログラム

i18n.c サンプル・プログラムは、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型

multthrd.c と *thrdfunc.c* サンプル・プログラム

このサンプル・プログラムは、マルチスレッド Client-Library アプリケーションの例を示します。このプログラムは、次の 2 つのファイルで構成されます。

- *multthrd.c* には、5 つのスレッドを生成するソース・コードが含まれています。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* には、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が含まれています。

プラットフォームが POSIX スレッドの完全な実装をサポートしていない場合は、このサンプル・プログラムを実行することはできません。「付録 B 環境変数」の説明に従って、SYBPLATFORM 環境変数を設定する必要があります。

rpc.c サンプル・プログラム

RPC コマンドのサンプル・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。

secct.c サンプル・プログラム

secct.c サンプル・プログラムは、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 Kerberos をインストールして稼働させる必要があります。また、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 (Adaptive Server や、Open Server の *secsrv.c* サンプル・プログラムなど) に接続することも必要です。

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 UNIX 版』を参照してください。

uni_blktxt.c サンプル・プログラム

uni_blktxt.c サンプル・プログラムは、バルク・コピー・ルーチンを使用して、unichar データ型や univarchar データ型などの静的データをサーバ・テーブルにコピーします。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 サンプル・プログラムは、計算結果を処理する方法を示します。このサンプル・プログラムは、unichar データ型と univarchar データ型を使用するために *compute.c* サンプル・プログラムを修正したものです。このサンプル・プログラムを実行するには、unipubs2 データベースが必要です。このプログラムは次のように動作します。

- 言語コマンドを使用してクエリをサーバに送信します。
- 標準の ct_results 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_fetch ループを使用してローをフェッチし、表示します。

unipubs2 データベースをインストールする方法については、*$SYBASE/$SYBASE_OCS/sample/ctlibrary* にある *README* ファイルを参照してください。

uni_csr_disp.c サンプル・プログラム

uni_csr_disp.c サンプル・プログラムは、読み込み専用カーソルの使い方を示します。このサンプル・プログラムは *csr_disp.c* サンプル・プログラムを修正したものです。実行するには unipubs2 データベースが必要です。このプログラムは次のように動作します。

- 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については、*$SYBASE/$SYBASE_OCS/sample/ctlibrary* にある *README* ファイルを参照してください。

uni_firstapp.c サンプル・プログラム

このプログラムは、unichar データ型と univarchar データ型を使用するように *firstapp.c* サンプル・プログラムを修正したものです。これは、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。*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。

uni_rpc.c サンプル・プログラム

RPC コマンドのサンプル・プログラム *uni_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・プログラムを修正したものです。実行するには unipubs2 データベースが必要です。unipubs2 データベースをインストールする方法については、*$SYBASE/$SYBASE_OCS/sample/ctlibrary* にある *README* ファイルを参照してください。

usedir.c サンプル・プログラム

usedir.c サンプル・プログラムは、使用可能なサーバのリストをディレクトリ・サービスに問い合わせる Client-Library の機能を示します。

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 *interfaces* 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。

ネットワーク・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 UNIX 版』を参照してください。

wide_compute.c サンプル・プログラム

wide_compute.c サンプル・プログラムは、ワイド・テーブルと大きなカラム・サイズを使用して計算結果を処理する方法を示します。このプログラムは次のように動作します。

- 言語コマンドを使用してクエリを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。

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。

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 compute 句によって返される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)
  ```

wide_curupd.c サンプル・プログラム

wide_curupd.c サンプル・プログラムは、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 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 サンプル・プログラムは、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。したがって、select 文を呼び出すたびに、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

RPC コマンドのサンプル・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。この動作は *rpc.c* プログラムと同じですが、ワイド・テーブルと大きなカラム・サイズを使用する点が異なります。

第 2 章

Open Client DB-Library/C

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。

Open Client DB-Library/C を使用できるオペレーティング・システム・プラットフォームのリストについては、『Open Server および SDK 新機能』(各 Windows、Linux、UNIX 版) を参照してください。

一般的な条件

サンプル・プログラムをはじめとする DB-Library アプリケーションを実行するには、以下の準備が必要です。

- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。
  - SYBASE
  - SYBASE_OCS
  - DSQUERY

  - SYBPLATFORM
  - プラットフォーム固有のライブラリ・パス変数
- Adaptive Server データベースに接続できる必要があります。『Open Client/Server 設定ガイド UNIX 版』を参照してください。
- *$SYBASE/$SYBASE_OCS/sample/dblibrary* 下の各製品のディレクトリにある *README* ファイルを読みます。サンプル・プログラムの実行方法の完全な手順については、*README* ファイルを参照してください。
- *sybopts.sh* ファイルの所有者に、このファイルに対する execute パーミッションを設定します。

```
chmod u+x sybopts.sh
```

DB-Library 実行プログラムの構築

ライブラリ、リンク、ヘッダ・ファイルを使用して、DB-Library 実行プログラムを構築します。

ライブラリ

すべてのプラットフォームに対応するライブラリを組み込むと、すべての DB-Library 機能を十分に活用できます。

- *libsybdb* － DB-Library (Sybase)
- *libsybunic* － Unicode-Library (Sybase)

コンパイルとリンク行

表 2-1 と 表 2-2 に、UNIX オペレーティング・システムが稼働し、Sybase がサポートするプラットフォーム上で、DB-Library アプリケーションのコンパイルとリンクを行うコマンドの一般的なフォーマットを示します。表 2-1 に、静的ライブラリを使用して DB-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 2-1: DB-Library の静的なコンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybdb -lsybunic -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybdb -lsybunic -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb -lsybunic -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybdb -lsybunic -Wl,-E,+s -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybdb -lsybunic -Wl,-E,+s -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb -lsybunic -ldl -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb -lsybunic -ldl -o program`
Linux x86-64 64 ビット版	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb64 -lsybunic64 -ldl -o program`

表 2-2 に、デバッグ・ライブラリを使用して DB-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 2-2: DB-Library のデバッグ・コンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -Wl,-E,+s -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -Wl,-E,+s -o program`
Linux x86 32 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -ldl -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -ldl -o program`
Linux x86-64 64 ビット版	`gcc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb64 -lsybunic64 -ldl -o program`

表 2-3 に、共有ライブラリをサポートするプラットフォーム上で DB-Library アプリケーションのコンパイルとリンク (動的ドライバを使用) を行うためのコマンドを示します。

表 2-3: DB-Library の共有コンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/$SYBASE_OCS.lib program.c -Bdynamic -lsybdb -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/$SYBASE_OCS.lib program.c -Bdynamic -lsybdb -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybdb -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybdb -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb -ldl -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb -ldl -o program`
Linux x86-64 64 ビット版	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb64 -ldl -o program`

パフォーマンスの考慮事項

共有ライブラリとリンクすると、静的ライブラリとリンクする場合よりも実行プログラムが小さくなり、リンク時間も少なくて済みます。ただし、共有ライブラリとリンクされた実行プログラムは、静的ライブラリを使用してリンクされた実行プログラムよりも起動に時間がかかります。さらに、静的ライブラリとは違って共有ライブラリは実行時に使用可能でなければなりません。

最高のパフォーマンスを提供するライブラリのタイプは、それぞれのサイトの稼働条件によって決まります。

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』参照) や、終了値 STDEXIT と ERREXIT などの記号定数を定義しています。*sybfront.h* ファイルには、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* ― 定義と型定義が追加されています。これらの定義のほとんどは DB-Library/C ルーチンだけが使用します。*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。

『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。

DB-Library のサンプル・プログラムの使用

DB-Library には、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 で提供されるサンプル・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・プログラムの目的

サンプル・プログラムは、DB-Library 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 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。実際の運用環境で使用できるレベルのプログラムでは、エラーや特殊なケースを処理するためのコードを追加する必要があります。

ロケーション

サンプル・プログラムは *$SYBASE/$SYBASE_OCS/sample/dblibrary* にあります。

このディレクトリには次のようなファイルが含まれています。

- サンプル・プログラムのソース・コード
- サンプル・プログラム用のデータ・ファイル
- サンプル・ヘッダ・ファイル *sybdbex.h*
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

$SYBASE/$SYBASE_OCS/sample/dblibrary の内容を、元のファイルの整合性に影響を与えないでサンプル・プログラムを自由に使用できるような作業ディレクトリにコピーしてから、コンパイルして実行してください。

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 *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));
```

例 5 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);
```

sybdbex.h 内の行に対して加えることができる変更は次のとおりです。

- 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変更します。
- PASSWORD は、*sybdbex.h* 内に null (" ") 文字列として定義されています。サンプル・プログラムを実行する前に、*sybdbex.h* を編集して "server_password" をサーバのパスワードに変更してください。PASSWORD について次のオプションのいずれかを選択します。

  - 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 "server_password" に変更します。このオプションは、セキュリティを侵害される可能性があります。パスワードがこのような公開された値に設定されていると、承認されていないユーザでもサーバにログインできるからです。このような場合は、次のいずれかを行います。
  - *sybdbex.h* 内の 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、DBSETLPWD 行全体を削除して、ユーザにその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。
- 使用するサーバの言語が英語でない場合は、*sybdbex.h* 内の LANGUAGE 行を編集して、サーバと同じ言語を指定します。LANGUAGE を参照するサンプル・プログラムは例 12 だけです。

サンプル・プログラムの概要

次のようなサンプル・プログラムがソフトウェアに含まれています。

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。

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。このサンプルを使用するには、提供されている *datafile* というファイルが必要です。また、ログイン・データベース内で create database パーミッションを持っていることを前提とします。

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ます。サンプル・プログラムは、集約結果と計算結果の両方のバインドの例を示すものです。

注意 このサンプル・プログラムを使用するには、Adaptive Server と pubs2 データベースにアクセスできる必要があります。

---

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。

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。このサンプル・プログラムはテーブルを作成して、そのテーブルにデータを挿入し、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 つずつ更新する必要があるアプリケーションに便利です。

注意 *example6.c* を使用するには、提供されている *datafile* というファイルが必要です。このプログラムはデフォルトのデータベース内に alltypes というテーブルを作成します。

---

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がどれであるかをアプリケーションで判断する方法を示します。また、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。

example8.c サンプル・プログラム

example8.c は、リモート・プロシージャ・コールを送信し、そのコール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・プログラムでは、デフォルト・データベース内にストアド・プロシージャ rpctest を作成しておかなければなりません。*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。

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。この手順は次のとおりです。

1　text 値または image 値以外のすべてのデータをそのローに挿入 (insert) します。

2　text または image の値を NULL に設定して、ローを更新 (update) します。null 値を持つ text カラムまたは image カラムのローに有効なテキスト・ポインタが含まれるのは、その null 値が update 文を使用して明示的に入力された場合だけであるので、この手順は重要です。

3　ローを選択 (select) します。text 値または image 値を含むカラムを明示的に選択 (select) します。この手順によって、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します。アプリケーションは、この select 文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、DBPROCESS からテキスト・ポインタを取り出します。dbtxtptr の *column* パラメータは、手順 3 で実行した select を参照する整数です。たとえば、次のような select を実行したとします。

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

ここでは、text_column は text カラムの名前です。dbtxtptr は、*column* パラメータが 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 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、まとまりごとに値を書き込む。

7 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結果として 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します。また、後で取り出せるように dbtxtsput を使用して保存します。

注意 このサンプル・プログラムを使用するには、pubs2 データベースが格納された Adaptive Server にアクセスできる必要があります。

---

example10.c サンプル・プログラム

example10.c は、著者 ID と image 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 image を読み込んで、著者 ID と image を含んでいる新しいローを pubs2 データベースの au_pix というテーブルに挿入します。text 値または image 値をデータベース・テーブルに挿入する方法については、example9.c を参照してください。

注意 このサンプル・プログラムを使用するには、pubs2 データベースが格納された Adaptive Server にアクセスできる必要があります。著者 ID の形式は 000-00-0000 でなければなりません。サンプル・コードと一緒に提供される *imagefile* ファイルには image が含まれています。

---

example11.c サンプル・プログラム

example11.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。ローを検索したあと、このサンプル・プログラムは pic カラムに含まれている image 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。

- このサンプル・プログラムでは、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。
- 他の方法として、dbmoretext とともに dbreadtext を使用して、text 値または image 値をさらに小さないくつかのデータ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・プログラムを使用するには、Adaptive Server と pubs2 データベースにアクセスできる必要があります。

example12.c サンプル・プログラム

example12.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・プログラムを使用するには、Adaptive Server と pubs2 データベースにアクセスできる必要があります。

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・プログラムを使用するには、Adaptive Server にアクセスできる必要があります。create database と create table パーミッションも必要です。

twophase.c サンプル・プログラム

twophase.c commit 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ら、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ます。

このサンプル・プログラムでは、SERVICE と PRACTICE という名前の 2 つのサーバ上で Adaptive Server が稼働していて、各サーバに pubs2 データベースが格納されていなければなりません。使用するサーバがこれとは違った名前である場合は、ソース・コード内の SERVICE と PRACTICE を実際に使用するサーバ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、クライアントが両方のサーバにアクセスできることを確認しておく必要があります。『Open Client/Server 設定ガイド UNIX 版』を参照してください。

注意 PRACTICE サーバが SERVICE サーバとは別のマシン上にある場合は、PRACTICE サーバから SERVICE のクエリ・ポートに接続できる必要があります。

第 3 章

Open Server Server-Library/C

Open Server Server-Library/C は、クライアント／サーバ・アーキテクチャの機能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。

Open Server Server-Library/C を使用できるオペレーティング・システム・プラットフォームのリストについては、『Open Server および SDK 新機能』(各 Windows、Linux、UNIX 版) を参照してください。

一般的な条件

サンプル・プログラムをはじめとする Open Server アプリケーションを実行するには、以下の準備が必要です。

- Adaptive Server と pubs2 サンプル・データベース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、『ASE インストール・ガイド』を参照してください。
- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。
  - SYBASE
  - SYBASE_OCS
  - DSQUERY と DSLISTEN
  - SYBPLATFORM
  - プラットフォーム固有のライブラリ・パス変数
- Adaptive Server に接続できる必要があります。『Open Client/Server 設定ガイド UNIX 版』を参照してください。
- *sybopts.sh* ファイルの所有者に、このファイルに対する execute パーミッションを設定します。

```
chmod u+x sybopts.sh
```

Server-Library 実行プログラムの構築

ライブラリ、リンク、ヘッダ・ファイルを使用して、Server-Library 実行プログラムを構築します。

ライブラリ

表 3-1 に、Server-Library のすべての機能を十分に活用するために組み込むライブラリを示します。表の 1 行目には、すべてのプラットフォームで使用するライブラリを示します。2 行目以降はプラットフォームに固有なライブラリを示しています。

表 3-1: プラットフォーム固有のライブラリ

プラットフォーム	必要なライブラリ
すべてのプラットフォーム	*libsybct* － Client-Library (Sybase) *libsybcs* － CS-Library (Sybase) *libsybtcl* － トランスポート制御層 (Sybase 内部使用) *libsybcomn* － 内部共有ユーティリティ・ライブラリ (Sybase 内部使用) *libsybintl* － 国際化サポート・ライブラリ (Sybase 内部使用) *libsybunic* － Unicode-Library (Sybase 内部使用) *libsybsrv* － Server-Library (Sybase) *libsybdb* － DB-Library (Sybase) *libm* － UNIX 標準の算術ライブラリ (システム)
Solaris プラットフォーム	*libthread* － スレッド・ライブラリ (システム) *libpthread* － スレッド・ライブラリ (システム) *libsocket* － ソケット・ネットワーク・ライブラリ (システム) *libnsl* － ネットワーク・ライブラリ (システム) *libdl* － 動的ローダ・ライブラリ (システム)
HP HP-UX プラットフォーム	*libcl* － HP トランスポート制御層 (システム) *libBSD* － BSD ライブラリ (システム) *libc_r* － C 言語リエントラント・ライブラリ *libdld* － 動的ローダ・ライブラリ (システム)
IBM AIX プラットフォーム	*libc_r* － C 言語リエントラント・ライブラリ *libpthreads* － スレッド・ライブラリ (システム)
Linux プラットフォーム	*libpthread* － スレッド・ライブラリ (システム)

コンパイルとリンク行のコマンド

以下の表は、UNIX オペレーティング・システムが稼働し、Sybase がサポートするプラットフォーム上で、Server-Library アプリケーションのコンパイルとリンクを行うコマンドの一般的なフォーマットを示します。

表 3-2 は、静的ライブラリを使用して Server-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 3-2: Server-Library の静的なコンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,shared_archive -lcl -lm -lBSD -Wl,-E,+s -o program`
HP HP-UX Itanium 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,shared_archive -lcl -lm -lBSD -Wl,-E,+s -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybsrv -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybsrv64 -lsybct64 -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -Wl,-Bdynamic -ldl -lnsl -lm64 -o program`

表 3-3 に、デバッグ・ライブラリを使用して Server-Library アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 3-3: Server-Library のデバッグ・コンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lnsl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lnsl -lm -lsocket -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb| -lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -o program`
HP HP-UX Itanium 32 ビット版および 64 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -o program`
Linux x86 32 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybsrv64 [-lsybdb64|-lsybct64] -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -ldl -lnsl -lm64 -o program`

表 3-4 に、共有ライブラリを使用して Server-Library アプリケーションのコンパイルとリンク (動的ドライバを使用) を行うためのコマンドを示します。

表 3-4: Server-Library の共有コンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/lib program.c -Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/lib program.c -Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/lib program.c -Wl,-Bdynamic -lsybsrv [-lsybdb| -lsybct] -lsybcs -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lcl -lm -lBSD -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lcl -lm -lBSD -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -ldl -lnsl -lm -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -ldl -lnsl -lm -o program`

プラットフォーム	コマンド
Linux x86-64 64 ビット版	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybsrv64 [ -lsybdb64 | -lsybct64 ] -lsybcs64 -ldl -lnsl -lm64 -o program`

注意 Open Server プログラムは、Client-Library ルーチンまたは DB-Library ルーチンを使用できます。上記の -lsybsrv の後の角カッコで囲まれている情報は、DB-Library 用の -lsybdb または Client-Library 用の -lsybct のどちらかを選択できることを示します。

表 3-5 は、スレッドセーフ・サポートを利用するために、Server-Library アプリケーションをコンパイルしてライブラリとリンクするためのコマンドを示します。

表 3-5: Server-Library のスレッドセーフのコンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_REENTRANT -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lnsl -ldl -lpthread -lthread -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_REENTRANT -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lnsl -ldl -lpthread -lthread -lm -lsocket -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc_r -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lpthread -lm -o program`

プラットフォーム	コマンド
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -D_REENTRANT -Ae -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lcl -lm -lBSD -lpthread -ldld -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -D_REENTRANT -Ae -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lcl -lm -lBSD -lpthread -ldld -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -ldl -lpthread -lnsl -lm -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc_r -q32 -g -D_REENTRANT -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -ldl -lpthread -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc_r -q32 -g -D_REENTRANT -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -lsybsrv64_r -lsybct64_r -lsybcs64_r -lsybtcl64_r -lsybcomn64_r -lsybintl64_r -ldl -lpthread -lnsl -lm64 -o program`

Kerberos サポート

Server-Library では、ネットワークを介して通信するときに高度なセキュリティを必要とするアプリケーションに対して Kerberos セキュリティ機能をサポートします。必要な Kerberos ソフトウェアをインストールし、適切な設定作業を実行することによって、Server-Library アプリケーションは次の Kerberos セキュリティ機能を利用できます。

- ネットワーク認証
- 相互認証
- 順序不整合認証

- リプレイの検出
- 機密性
- 整合性

表 3-6: Kerberos サポートに必要な作業

タスク	参照先
次の Kerberos ソフトウェアをシステムにインストールする。*GSS* ライブラリが共有ライブラリとしての使用をサポートしていることを確認する。	Kerberos のマニュアルおよび『Open Client/Server 設定ガイド UNIX 版』を参照。
Kerberos ユーティリティ kadmin. を使用して、キー・テーブル・ファイルに必要なサーバ・プリンシパル用のキーを設定する。	Kerberos のマニュアルを参照。
libtcl.cfg 設定ファイルのセキュリテイ・セクションを設定します。	『Open Client/Server 設定ガイド UNIX 版』を参照してください。
Client-Library アプリケーションを Sybase リエントラント・ライブラリにリンクする。	「Kerberos サポート」(52 ページ) を参照してください。

• CyberSafe Kerberos の場合
  • CSFC5CCNAME 環境変数にクレデンシャル・キャッシュ・ディレクトリのロケーションを設定する。
  • CSFC5KTNAME 環境変数にキー・テーブル・ファイルのパスを設定する (デフォルト・キー・テーブル・ファイル以外を使用する場合)。
• MIT Kerberos の場合
  • KRB5CCNAME 環境変数にクレデンシャル・キャッシュ・ファイルのロケーションを設定する。
  • KRB5_KTNAME 変数にキー・テーブル・ファイルのパスを設定する (デフォルトのキー・テーブル・ファイル以外を使用する場合)。
	Kerberos のマニュアルを参照。
	クレデンシャル・キャッシュ・ディレクトリのデフォルト・ロケーションはプラットフォームごとに異なる。
	• CyberSafe Trust Broker の場合、デフォルトのキー・テーブル・ファイルは */krb5/v5srvtab*。
	• MIT Kerberos の場合、デフォルトのキー・テーブル・ファイルは */etc/krb5.keytab*。

サーバ・プリンシパル名が srv_init に渡されるサーバ名と異なる場合、srv_props を使用してサーバ・プリンシパル名を設定する。	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。

注意 セキュリティを強化するために、Sybase では Open Server を実行するユーザ ID を *key table* ファイルの所有者にし、他のすべてのユーザによるこのファイルへのアクセスを制限することをおすすめします。さらに、各 Open Server を実行するときには、対話型処理に使用されないユニークなユーザ ID を使用してください。

バルク・コピー・ルーチン

バルク・コピー・ルーチンを使用する場合は、libblk バルク・コピー・ライブラリにリンクする必要があります。リンク・コマンド行の `-lsybsrv` の前に、`-llsybblk` を追加します。

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

パフォーマンスの考慮事項

共有ライブラリとリンクすると、静的ライブラリとリンクする場合よりも実行プログラムが小さくなり、リンク時間も少なくて済みます。ただし、共有ライブラリとリンクされた実行プログラムは、静的ライブラリを使用してリンクされた実行プログラムよりも起動に時間がかかります。さらに、静的ライブラリとは違って共有ライブラリは実行時に使用可能でなければなりません。

最高のパフォーマンスを提供するライブラリのタイプは、サイトの個々の稼働条件によって決まります。

ヘッダ・ファイル

ospublic.h ヘッダ・ファイルは、すべての Open Server アプリケーションのソース・ファイルにインクルードする必要があります。他の必要なヘッダ・ファイルは、*ospublic.h* 内にネストされています。Bulk-Library を使用する場合は、*ospublic.h* の他に *bkpublic.h* もインクルードしてください。

ヘッダ・ファイルの詳細については、『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。

Server-Library のサンプル・プログラムの使用

この項では、Server-Library に付属しているサンプル・プログラムについて説明します。

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い方を示します。これらのサンプル・プログラムはサーバであるため、*interfaces* ファイル内またはネットワーク・ディレクトリ・サービス内にそのマシンとネットワーク・アドレスを示すエントリが必要です。ディレクトリ・サービスの設定方法については、『Open Client/Server 設定ガイド UNIX 版』を参照してください。

サンプル・プログラムの目的

サンプル・プログラムは、Open Server に固有の機能の例を示します。これらのプログラムは、ctos 以外は、Open Server 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。実際の運用環境で使用できるレベルのプログラムでは、エラーや特殊なケースを処理するためのコードを追加する必要があります。

これらのサンプル・プログラムで使用できるトレース・フラグについては、それぞれのサンプル・プログラムで確認してください。サンプル・プログラムを実行する方法の詳細については、*README* ファイルを参照してください。

ロケーション

サンプル・プログラムは、*$SYBASE/$SYBASE_OCS/sample/srvlibrary* にあります。このディレクトリには次のファイルが含まれています。

- サンプル・プログラムのソース・コード

- サンプル・プログラムを構築するための *makefile*。このサンプル・プログラムは、Server-Library アプリケーションの作成を開始するときに使用します。
- サンプル・ヘッダ・ファイル *ossample.h*.
- srv_connect イベント・ハンドラ
- エラー・ハンドラ
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

サンプル・プログラムの概要

次のようなサンプル・プログラムがソフトウェアに含まれています。

注意 シングルスレッドおよびマルチスレッドのサンプル・プログラムの場合、クライアントは stop_serv レジスタード・プロシージャを使用してサンプル・プログラムを停止します。

---

ctos.c サンプル・プログラム

ctos.c は、Server-Library 呼び出しと Client-Library 呼び出しを使用する Open Server ゲートウェイ・アプリケーションです。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
- オプション・コマンド
- RPC (リモート・プロシージャ・コール)

さらに、ctos.c は 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。

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。

dynlisten.c サンプル・プログラム

dynlisten.c サンプル・プログラムは、稼動中の Open Server がワイルドカードの IP ポート番号で新しいリスナを起動し、ログインのリダイレクトや接続のマイグレートに使用できる最終アドレス情報を取得する方法を示します。

exfds.c サンプル・プログラム

exfds.c は、Open Server のプロセス全体をブロックすることなく Open Server アプリケーションが外部ファイル記述子にサービスを提供できる方法を示します。このサンプル・プログラムでは、次の処理を行います。

- srv_capability ルーチンを使用して、現在のプラットフォームが srv_poll をサポートしているかどうかを調べます。
- 2 つの UNIX パイプをオープンします。
- srv_spawn ルーチンを使用して、srv_poll と srv_stop の 2 つのサービス・スレッドを生成します。

この 2 つのサービス・スレッドは、UNIX パイプにメッセージを書き込むことによって簡単なコマンドとその応答プロトコルを実装します。srv_poll を使用すると、メッセージを待っている間に Open Server がサービス・スレッドを再スケジュールできるようになります。進行状況をモニタする情報が *srv.log* に書き込まれます。Open Server は、ソース・コードで指定された回数だけコマンドとその応答プロトコルを実行してから、SRV_STOP イベントをキューイングします。

このサンプル・プログラムはクライアント・アプリケーションを必要としません。プログラムが正しく起動したかどうか調べるには、*srv.log* ファイルのメッセージを確認してください。

fullpass.c サンプル・プログラム

fullpass.c は、Sybase 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。『Open Server Server-Library/C リファレンス・マニュアル』の「第 2 章 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。

halang.c サンプル・プログラム

halang.c は、フェールオーバ後に Open Client が Open Server に再接続できるように HA セッション ID を設定する方法を示します。このサンプル・プログラムを使用して、Open Server で HA 機能を実装します。

サンプル・プログラムを実行するには、halang *your_open_server_name* -s *failover_server* & コマンドを実行します。

注意 Open Server アプリケーションによって HA セッション ID が生成されます。

---

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を示します。このプログラムは、Open Server アプリケーションの各国言語と文字セットのための値を初期設定してから、クライアントの要求に応じ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。

lang.c サンプル・プログラム

lang.c は、srv_language イベント・ハンドラの使用方法を示します。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。

言語コマンドを処理する方法の詳細については、『Open Server Server-Library/C リファレンス・マニュアル』の「言語呼び出し」の項を参照してください。

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)

サービス・スレッドは、この 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。

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)。ロガーは連続的にループを繰り返してメッセージを待ちます。メッセージを受け取ると、クエリの内容をファイルに出力して、送信側をウェイクアップします。

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。*multthrd.c* プログラムは、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。

osintro.c サンプル・プログラム

osintro.c は、Open Server アプリケーションの基本的なコンポーネントの例を示します。イベント・ハンドラはインストールされていません。

mqueue.c サンプル・プログラム

mqueue.c サンプル・プログラムは、srv_createmsgq()、srv_putmsgq()、srv_getmsgq() の各 API 関数を使用した Open Server のメッセージ・キューの使用例を示します。サーバに送られるすべての言語コマンドは、言語ハンドラによってメッセージ・キューに格納されます。サービス・スレッド・ロガーがキューからメッセージを読み取り、それを stdout に表示し、ログに格納します。

paramreader.c サンプル・プログラム

paramreader.c サンプル・プログラムは、単純なスタンドアロンの Open Server a アプリケーションの例を示すものです。このサンプル・プログラムでは、SRV_RPC および SRV_DYNAMIC イベント・ハンドラをインストールして、クライアント・アプリケーションから受け取る受信 (LOB) パラメータを表示します。このようなパラメータを送信するために、クライアント・サンプル・プログラムの *lobrpc.c* と *lobdynamic.c* が提供されています。これらのサンプル・プログラムは、主に、RPC または 動的 SQL で TEXT、IMAGE、または UNITEXT 型のパラメータを送受信する方法を示すために提供されています。

redirect.c サンプル・プログラム

redirect.c は、Open Client が interfaces ファイルでもともと指定されていたサーバとは異なるサーバにログインするようにする、単純な Open Server アプリケーションです。15.0 以降のクライアントでのみ動作します。

サンプル・プログラムを実行するには、redirect *your_open_server_name* -s *alternate_server_to_use* コマンドを実行します。

注意 目的のサーバは、ログイン・パケット内の情報 (アプリケーション名フィールドやサーバ名フィールドなど) に基づくことができます。クライアントは、リダイレクト・サーバによって提供される接続情報を使用して、そのログイン試行を再開します。

---

regproc.c サンプル・プログラム

regproc.c は、Open Server でのレジスタード・プロシージャの使用例を示します。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、クライアント・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 で使用するために、次のクライアント・プログラムが追加されています。

- *version.c* － Open Server のバージョンを返すレジスタード・プロシージャ (rp_version) を実行します。
- *dbwait.c* － DB-Library とともに実装され、rp_version が実行されるときに Open Server が通知を受けるように登録します。
- *ctwait.c* － Client-Library とともに実装され、rp_version が実行されるときに Open Server が通知を受けるように登録します。

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。

『Open Client/Server 設定ガイド UNIX 版』を参照してください。

sendrpc.c サンプル・プログラム

sendrpc.c サンプル・プログラムは、単純な RPC コマンドを Adaptive Server または Open Server アプリケーションに送信し、返された結果を処理する方法を示します。このサンプル・プログラムでは、パラメータのない単純な RPC コマンドのみを使用します。sp_who や ctos_shutdown がその例です。

sigalarm.c サンプル・プログラム

sigalarm.c は、Open Server アプリケーションが UNIX の SIGALARM シグナルを使用して周期的なイベントをスケジュールする方法を示します。*sigalarm.c* は具体的には次のような処理を行います。

- srv_spawn を使用して、アラームによってウェイクアップされるまでスリープするサービス・スレッドを生成します。サービス・スレッドは、ウェイクアップされるたびに srv_log ルーチンを使用して、Open Server ログ・ファイルにメッセージを書き込みます。
- srv_signal ルーチンを使用して、SIGALARM ハンドラが呼び出されるたびにスリープ中のサービス・スレッドをウェイクアップする SIGALARM ハンドラをインストールします。*sigalarm.c* は、UNIX の alarm を呼び出すことによって、特定の間隔で SIGALARM が配信されるように要求します。

このサンプル・プログラムはクライアント・アプリケーションを必要としません。プログラムが正しく起動したかどうか調べるには、*srv.log* ファイルのメッセージを確認してください。

timedsleep.c サンプル・プログラム

timedsleep.c サンプル・プログラムは、srv_timedsleep()、srv_createmutex()、srv_lockmutex()、srv_unlockmutex() の各 API 関数の使用方法を示します。このサンプル・プログラムを使用して、2 つの isql 接続でじゃんけんゲームをします。

注意 srv_timedsleep() API 関数とこのサンプル・プログラムは、スレッド・ライブラリでのみ使用できます。

---

updtext.c サンプル・プログラム

updtext.c サンプル・プログラムは、*uctext ctlibrary* サンプル・プログラムとともに使用します。このサンプル・プログラムでは、SRV_LANGUAGE イベント・ハンドラをインストールし、受信した言語コマンドが含まれる情報メッセージを使用してクライアントのコマンドに応答します。

また、このサンプル・プログラムでは、受信したテキストを示すメッセージをクライアントに送信する、バルク・ハンドラもインストールします。これは updatetext 機能を示すものです。サーバに対してクエリを実行してこの機能がサポートされているかどうかを確認できるため、rpc_handler がインストールされています。rpc_handler は sp_mda rpc を確認します。この rpc は、サーバに関するメタデータ (特に text の部分更新のサポート) を使用してクライアントに応答します。

第 4 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-SQLR 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、修正を行うための簡単な方法を提供します。

Open Client Embedded SQL/C を使用できるオペレーティング・システム・プラットフォームのリストについては、『Open Server および SDK 新機能』(各 Windows、Linux、UNIX 版) を参照してください。

一般的な条件

サンプル・プログラムをはじめとする Embedded SQL/C アプリケーションを実行するには、以下の準備が必要です。

- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。
  - SYBASE
  - SYBASE_OCS
  - SYBPLATFORM
  - プラットフォーム固有のライブラリ・パス変数

- 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 にアクセスできるようにします。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。
- *sybopts.sh* ファイルの所有者に、このファイルに対する execute パーミッションを設定します。

  ```
  chmod u+x sybopts.sh
  ```

- 検索パスに現在のディレクトリを追加する (まだ指定していない場合)。

  ```
  setenv PATH .:$PATH
  ```

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て、必要なファイルやライブラリとアプリケーションをリンクします。

3 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

ソース・プログラムをプリコンパイルするコマンドのフォーマットは次のとおりです。

```
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]]
```

```
[-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]
```

filename は、Embedded SQL/C ソース・ファイルの名前です。*filename* のデフォルトの拡張子は、".cp" です。cpre は、".c" 拡張子の付いた出力ファイルを生成します。

注意 cpre64 と cpre_r64 は、64 ビット・アプリケーション用のプリコンパイラです。cpre64 プリコンパイラは、非リエントラント・プリコンパイラであり、cpre_r64 はリエントラント・バージョンです。これらのプリコンパイラは、Open Client と Open Server でサポートされるすべての 64 ビット・プラットフォームで使用できます。

オプションの一部には、プリコンパイラの機能を有効にするためのスイッチもあります。たとえば、あるオプションはストアド・プロシージャを生成します。デフォルトでは、これらの機能はオフになっています。オンにするには、cpre コマンド行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)。

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。

プリコンパイラ・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

ライブラリ、リンク、ヘッダ・ファイルを使用して、Embedded/SQL C アプリケーションをコンパイルしてリンクします。

Client-Library と Server-Library は、Net-Library™ ドライバ、ディレクトリ・ドライバ、セキュリティ・ドライバの動的ロードをサポートしています。次の Sybase オブジェクト・ファイルは、アプリケーションと明示的にリンクする必要はありません。

- Net-Library ドライバ
- ディレクトリ・ドライバ
- セキュリティ・ドライバ

以下の表に、UNIX 上で稼働し、Sybase がサポートするプラットフォーム上で、Embedded SQL/C アプリケーションのコンパイルとリンクを行うためのコマンドの一般的なフォーマットを示します。

表 4-1 は、静的ライブラリを使用して Embedded SQL/C アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 4-1: Embedded SQL/C の静的なコンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib` *`gen_program.c`* `$SYBASE/$SYBASE_OCS/include/sybesql.c -Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib` *`gen_program.c`* `$SYBASE/$SYBASE_OCS/include/sybesql.c -Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-a,archive -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,default -lcl -lm -lBSD -ldld -Wl, -E, +s -o program`

プラットフォーム	コマンド
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-a,archive -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,default -lcl -lm -lBSD -ldld -Wl, -E, +s -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl, -Bstatic -lsybct64 -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -Wl, -Bdynamic -ldl -lnsl -lm64 -o program`

表 4-2 は、デバッグ・ライブラリを使用して Embedded SQL/C アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 4-2: Embedded SQL/C のデバッグ・コンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`/opt/SUNWspro/bin/cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`

プラットフォーム	コマンド
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -ldld -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -ldld -o program`
Linux x86 32 ビット版	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct64 -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -ldl -lnsl -lm64 -o program`

表 4-3 に、共有ライブラリを使用して Embedded SQL/C アプリケーションのコンパイルとリンク (動的ドライバを使用) を行うためのコマンドを示します。

表 4-3: Embedded SQL/C の共有コンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Bdynamic -lsybct -lsybcs -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Bdynamic -lsybct -lsybcs -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bdynamic -lsybct -lsybcs -lm -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,a,shared_archive -lsybct -lsybcs -lcl -lm -lBSD -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,a,shared_archive -lsybct -lsybcs -lcl -lm -lBSD -o program`
Linux x86 32 ビット版	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux POWER 32 ビット版および 64 ビット版	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux x86-64 64 ビット版	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bdynamic -lsybct64 -lsybcs64 -ldl -lnsl -lm64 -o program`

注意 *sybesql.c* ファイルをコンパイルすることによって生成されるオブジェクトには、Embedded SQL/C アプリケーションで使用されるユーティリティ・ルーチンが含まれます。アプリケーションが正しく動作するためには、すべてのアプリケーションに *sybesql.o* をリンクしなければなりません。

- Embedded SQL/C アプリケーションのリンク行は、Client-Library アプリケーションの場合に使用するリンク行と同一です。リンク行の *gen_program.c* は、cpre から生成された C ファイルです。
- -lsybct は、コードが呼び出す Open Client ライブラリをリンクするためのリンカ・オプションを表します。-lsybct の他にも、次に示すリンカ・オプションの一部またはすべてをこの順序で指定できます。

非スレッド・アプリケーションの場合	スレッド・アプリケーションの場合
-lsybsrv (Server-Library ルーチン用)	-lsybsrv_r (Server-Library ルーチン用)
-lsybblk (Bulk-Library ルーチン用)	-lsybblk_r (Bulk-Library ルーチン用)
-lsybct (Client-Library ルーチン用)	-lsybct_r (Client-Library ルーチン用)

- 64 ビット C アプリケーションを構築するには、-DSYB_LP64 コンパイラ・オプションを使用して、C コンパイラが正しいコードを生成していることを確認します。*$SYBASE/$SYBASE_OCS/sample/esqlc* に用意されている sybopts.sh スクリプトを参照してください。

HP HP-UX システムのユーザの場合

- -Wl,-a,archive オプションを使用すると、リンカは Sybase ライブラリを静的にリンクします。このオプションを指定していない場合は、Sybase ライブラリの共有バージョンが使用されます。この場合、実行時に SH_LIB_PATH 環境変数に *$SYBASE/$SYBASE_OCS/lib* を含める必要があります。また、アプリケーション・ユーザには、*$SYBASE/$SYBASE_OCS/lib* 内のライブラリに対する read と execute のパーミッションが必要です。

- アプリケーションが +s リンカ・オプションを使用してリンクされている場合を除き、HP HP-UX は、実行時に SH_LIB_PATH 環境変数を使用しません。システムが実行時に Sybase ライブラリを見つけることができるようにするには、+s リンカ・オプションを使用してください。-E は、実行時にドライバ・ライブラリがロードされるときに、未定義シンボル・エラーにならないようにするために必要です。「HP-UX ld」を参照してください。

その他の注意事項

アプリケーションのコンパイルとリンクを行うときは、パフォーマンスと整列に関する以下の点を考慮してください。

パフォーマンス

共有ライブラリとリンクすると、静的ライブラリとリンクする場合よりも実行プログラムが小さくなり、リンク時間も少なくて済みます。ただし、共有ライブラリとリンクされた実行プログラムは、静的ライブラリを使用してリンクされた実行プログラムよりも起動に時間がかかります。さらに、静的ライブラリとは違って共有ライブラリは実行時に使用可能でなければなりません。

最高のパフォーマンスを提供するライブラリのタイプは、それぞれのサイトの稼働条件によって決まります。

64 ビット・アーキテクチャでのデータの整列

64 ビット・アプリケーションを構築する場合、データ構造体は 8 バイト境界 (8 バイトの倍数であるメモリ・アドレス) に整列させる必要があります。同様に、32 ビット・アプリケーションのデータ構造体は、4 バイト境界に整列させなければなりません。

ストアド・プロシージャのロード

プリコンパイラの -G フラグを使用してストアド・プロシージャを生成する場合は、isql を使用してストアド・プロシージャを Adaptive Server にロードしてから、プログラムを実行する必要があります。生成されたスクリプトを実行するための isql コマンドのフォーマットは、次のとおりです。

```
isql -Uuserid -Ppassword < program.sql
```

-U フラグには Adaptive Server にログインするためのユーザ ID を指定し、-P フラグにはパスワードを指定します。

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 のサンプル・プログラムの使用

Embedded SQL/C プリコンパイラには、一般的な Embedded SQL/C アプリケーション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・プログラムの目的

サンプル・プログラムは、Embedded SQL/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 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。

サンプル・プログラムを実行する方法の詳細については、*README* ファイルを参照してください。

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。実際の運用環境で使用できるレベルのプログラムでは、エラーや特殊なケースを処理するためのコードを追加する必要があります。

ロケーション

サンプル・プログラムは、*$SYBASE/$SYBASE_OCS/sample/esqlc* にあります。

このディレクトリには次のファイルが含まれています。

- サンプル・プログラムのソース・コード

- サンプル・プログラムを構築するための *makefile*。*makefile* は、Embedded SQL アプリケーションの作成を開始するときに使用します。
- サンプル・ヘッダ・ファイル *sybsqlex.h.*
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

sybopts.sh ファイルの所有者に、このファイルに対する execute パーミッションを設定します。

```
chmod u+x sybopts.sh
```

$SYBASE/$SYBASE_OCS/sample/esqlc の内容を、元のファイルの整合性に影響を与えないでサンプル・プログラムを自由に使用できるような作業ディレクトリにコピーしてから、コンパイルして実行してください。

ヘッダ・ファイル

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 に有効な値で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。

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    *
 *examples                                        *
 *                                                *
 **************************************************/

 #define USER     "username"
 #define PASSWORD "password"
 #define ERREXIT -1
 #define STDEXIT  0
```

すべてのサンプル・プログラムは次の行を含んでいます。

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

sybsqlex.h 内では、USER は "user name"、PASSWORD は "password" 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 *sybsqlex.h* を編集して "user name" を Adaptive Server のログイン名に置き換え、"password" を Adaptive Server のパスワードに置き換えてください。

example1.cp サンプル・プログラム

example1.cp は、対話型クエリ・プログラムでのカーソルの使い方を示します。この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て、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著者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した後、データを Adaptive Server に送ります。

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 コード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ものとほぼ同じです。フェールオーバの検出と処理には、エラー・ハンドラが使用されます。

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実行する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ものとほぼ同じです。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ものとほぼ同じです。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 5 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。

Open Client Embedded SQL/COBOL を使用できるオペレーティング・システム・プラットフォームのリストについては、『Open Server および SDK 新機能』(各 Windows、Linux、UNIX 版) を参照してください。

一般的な条件

サンプル・プログラムをはじめとする Embedded SQL/COBOL アプリケーションを実行するには、以下の準備が必要です。

- 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 にアクセスできるようにします。Adaptive Server Enterprise の『インストール・ガイド』を参照してください。
- 次の環境変数を設定します。詳細については、「付録 B 環境変数」を参照してください。

- SYBASE
- SYBASE_OCS
- COBDIR
- PATH
- SYBPLATFORM
- プラットフォーム固有のライブラリ・パス変数

Embedded SQL/COBOL 実行プログラムの構築

ライブラリ、リンク、ヘッダ・ファイルを使用して、Embedded SQL/COBOL 実行プログラムを構築します。

ライブラリ

表 5-1 に、Embedded SQL/COBOL のすべての機能を十分に活用するために組み込むライブラリを示します。表の 1 行目には、すべてのプラットフォームで使用できるライブラリを示します。2 行目以降は各プラットフォームに固有なライブラリを示します。

表 5-1: Embedded SQL/COBOL 用のプラットフォーム固有ライブラリ

プラットフォーム	サポートされているライブラリ
すべてのプラットフォーム	*libsybcobct* − Client-Library と CS-Library に対する COBOL インタフェース (Sybase) *libsybct* − Client-Library (Sybase) *libsybcs* − CS-Library (Sybase) *libsybunic* − Unicode-Library (Sybase 内部使用) *libsybcomn* − 内部共有ユーティリティ・ライブラリ (Sybase 内部使用) *libsybintl* − 国際化サポート・ライブラリ (Sybase 内部使用) *libsybtcl* − トランスポート制御層 (Sybase 内部使用) *libm* − UNIX 標準の算術ライブラリ (システム)
Solaris プラットフォーム	*libpthread* − スレッド・ライブラリ (システム) *libsocket* − ソケット・ネットワーク・ライブラリ (システム) *libnsl* − ネットワーク・ライブラリ (システム) *libdl* − 動的ローダ・ライブラリ (システム) *libthread* − ネイティブ・スレッド・ライブラリ (システム)

プラットフォーム	サポートされているライブラリ
HP HP-UX PA-RISC	*libsybcobct.sl* － 32 ビット動的共有ベニア層ライブラリ *libsybcobct_r.sl* － 32 ビット動的共有ベニア層ライブラリ (リエントラント・バージョン) *libsybcobct64.sl* － 64 ビット動的共有ベニア層ライブラリ *libsybcobct_r64.sl* － 64 ビット動的共有ベニア層ライブラリ (リエントラント・バージョン)
HP HP-UX Itanium	*libcl* － HP トランスポート制御層 (システム) *libBSD* － BSD ライブラリ (システム) *libc_r* － C 言語リエントラント・ライブラリ *libdld* － 動的ローダ・ライブラリ (システム)
IBM AIX RS/6000	*libc_r* － C 言語リエントラント・ライブラリ *libpthread* － スレッド・ライブラリ (システム)
Linux x86 32 ビット版	*libsybcobct* － Client-Library と CS-Library に対する COBOL インタフェース (Sybase) *libsybct* － Client-Library (Sybase) *libsybcs* － CS-Library (Sybase) *libsybtcl* － トランスポート制御層 (Sybase 内部使用) *libsybcomn* － 内部共有ユーティリティ・ライブラリ (Sybase 内部使用) *libsybintl* － 国際化サポート・ライブラリ (Sybase 内部使用) *libsybunic* － Unicode-Library (Sybase 内部使用) *libdl* － 動的ローダ・ライブラリ (システム) *libnsl* － ネットワーク・ライブラリ (システム) *libm* － UNIX 標準の算術ライブラリ (システム)
ESQL/COBOL をサポートする他のすべてのプラットフォーム	*libsybcobct.so* － 32 ビット動的共有ベニア層ライブラリ *libsybcobct_r.so* － 32 ビット動的共有ベニア層ライブラリ (リエントラント・バージョン) *libsybcobct64.so* － 64 ビット動的共有ベニア層ライブラリ *libsybcobct_r64.so* － 64 ビット動的共有ベニア層ライブラリ (リエントラント・バージョン)

Embedded SQL/COBOL アプリケーションから実行プログラムを構築する場合、次の 3 つの基本手順があります。

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およびリンクします。

3 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/COBOL ソース・プログラムをプリコンパイルするコマンドのフォーマットは、次のとおりです。

```
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]
```

filename は、Embedded SQL/COBOL ソース・ファイルの名前です。*filename* のデフォルトの拡張子は ".pco" です。cobpre を実行すると、".cbl" 拡張子の付いた出力ファイルが生成されます。

注意 cobpre64 と cobpre_r64 は、64 ビット・アプリケーション用のプリコンパイラです。cobpre64 は、非リエントラント・プリコンパイラであり、cobpre_r64 はリエントラント・バージョンです。

オプションの一部には、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するためのスイッチもあります。デフォルトでは、これらの機能はオフになっています。オンにするには、cobpre コマンド行でオプションを指定します。この他のコマンド修飾子は、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)。

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。

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

以下の表は、UNIX オペレーティング・システムが稼働し、Sybase がサポートするプラットフォーム上で、Embedded SQL/COBOL アプリケーションのコンパイルとリンクを行うコマンドの一般的なフォーマットを示します。

表 5-2 に、非デバッグ・ライブラリを使用して Embedded SQL/COBOL アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 5-2: Embedded SQL/COBOL の静的なコンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lBSD -lcl -lm -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic -lcl -lm -ldld -o program`

プラットフォーム	コマンド
IBM AIX RS/6000 32 ビット版および 64 ビット版	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
Linux x86 32 ビット版	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic-ldl -lnsl -lm -o program`

表 5-3 は、デバッグ・ライブラリを使用して Embedded SQL/COBOL アプリケーションのコンパイルとリンクを行うためのコマンドを示します。

表 5-3: Embedded SQL/COBOL のデバッグ・コンパイルとリンクのコマンド

プラットフォーム	コマンド
Solaris SPARC 32 ビット版および 64 ビット版	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32 ビット版および 64 ビット版	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
HP HP-UX PA-RISC 32 ビット版および 64 ビット版	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -o program`
HP HP-UX Itanium) 32 ビット版および 64 ビット版	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic -lcl -lm -ldld -o program`
IBM AIX RS/6000 32 ビット版および 64 ビット版	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`

プラットフォーム	コマンド
Linux x86 32 ビット版	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic-ldl -lnsl -lm -o program`

64 ビット COBOL アプリケーションを構築するときは、COBOL コンパイラのビルド・モードが正しく設定されていることを確認してください。たとえば、ESQL/COBOL では、COBMODE 環境変数を 32 ビット・ビルドでは 32、64 ビット・ビルドでは 64 に設定する必要があります。この設定を行わないと、ビルド・エラーになることがあります。また、32 ビットと 64 ビットの両方の COBOL アプリケーションをサポートするプラットフォーム上で、予期しないシグニチャが付いた実行プログラムが生成される場合もあります。64 ビット・アプリケーションの構築とリンクの詳細については、*SYBASE/$SYBASE_OCS/sample/esqlcob* にある sybopts.sh スクリプトを参照してください。

その他の注意事項

アプリケーションのコンパイルとリンクを行うときは、整列に関する以下の点を考慮してください。

64 ビット・アーキテクチャでのデータの整列

64 ビット・アプリケーションを構築する場合、データ構造体を 8 バイト境界 (8 バイトの倍数であるメモリ・アドレス) に整列させる必要があります。同様に、32 ビット・アプリケーションのデータ構造体は、4 バイト境界に整列させなければなりません。

以下の例では、32 ビットと 64 ビットの ESQL/COBOL バージョンの SQLDA を作成することで、この概念について説明しています。SQLDA は、動的 SQL で参照されるオブジェクトを記述する記述子領域です。以下の例では、C 言語で作成された Sybase バージョンの SQLDA を参考として提供しています。

例

このコード・スニペットは、Sybase が提供する SQLDA のレイアウトを示しています。

```
typedef struct _sqlda
{
   CS_SMALLINT sd_sqln;
```

```
      CS_SMALLINT sd_sqld;
      struct _sd_column
      {
         CS_DATAFMT sd_datafmt;
         CS_VOID *sd_sqldata;
         CS_SMALLINT sd_sqlind;
         CS_INT sd_sqllen;
         CS_VOID *sd_sqlmore;
      } sd_column[1];

   } syb_sqlda;

   typedef syb_sqlda SQLDA;
```

次の SQLDA 構造体は、Sybase 固有の SQLDA の 32 ビット ESQL/COBOL バージョンを示しています。

```
   01 OUT-DES. /* 32bit */
      09 SD-SQLN PIC S9(4) COMP.
      09 SD-SQLD PIC S9(4) COMP.
      09 SD-COLUMN OCCURS 27 TIMES./* 27-column table*/
         19 SD-DATAFMT.
            29 SQL--NM PIC X(256).
            29 SQL--NMLEN PIC S9(9) COMP.
            29 SQL--DATATYPE PIC S9(9) COMP.
            29 SQL--FORMAT PIC S9(9) COMP.
            29 SQL--MAXLENGTH PIC S9(9) COMP.
            29 SQL--SCALE PIC S9(9) COMP.
            29 SQL--PRECISION PIC S9(9) COMP.
            29 SQL--STTUS PIC S9(9) COMP.
            29 SQL--COUNT PIC S9(9) COMP.
            29 SQL--USERTYPE PIC S9(9) COMP.
            29 SQL--LOCALE PIC S9(9) COMP.
         19 SD-SQLDATA PIC S9(9) COMP.
         19 SD-SQLIND  PIC S9(4) COMP.
         19 FILLER PIC S9(4) COMP. /* Filler record to */
                                  /* align SQLIND   */
         19 SD-SQLLEN  PIC S9(9) COMP.
         19 SD-SQLMORE PIC S9(9) COMP.
```

Sybase 固有の SQLDA の 32 ビット ESQL/COBOL バージョンで、PICTURE (PIC) 句に関連する要素は次のとおりです。

- S9(4) と定義されている要素 － S9(4) は、ESQL/COBOL の 2 バイト長の smallint に相当します。S9(4) と定義されている要素は、単独では 32 ビットのデータ整列要件を満たしていません。ただし、SD-SQLN と SD-SQLD のように、2 つの要素をペアにすると、4 バイトの倍数であるメモリ・アドレスを占めることになるため、S9(4) のペアであればこの要件を満たします。
- S9(9) と定義されている要素 － S9(9) は、ESQL/COBOL の 4 バイト長の int に相当します。S9(9) と定義されている要素は、32 ビットのデータ整列要件を満たします。
- FILLER － ペアになっていない S9(4) 要素である SD-SQLIND を埋め込み、構造体全体を 4 バイト境界に配置するために、2 バイト長のフィラー・レコードが追加されます。

次の SQLDA 構造体は、Sybase 固有の SQLDA の 64 ビット ESQL/COBOL バージョンを示しています。64 ビット環境では、データ構造体全体が 8 バイト境界に整列する必要があります。

```
01 OUT-DES. /* 64 bit */
   09 SD-SQLN PIC S9(4) COMP.
   09 SD-SQLD PIC S9(4) COMP.
   09 FILLER PIC S9(9) COMP. /* First filler to align */
                             /* on eight bytes */
   09 SD-COLUMN OCCURS 27 TIMES./* 27-column table*/
      19 SD-DATAFMT.
         29 SQL--NM PIC X(256).
         29 SQL--NMLEN PIC S9(9) COMP.
         29 SQL--DATATYPE PIC S9(9) COMP.
         29 SQL--FORMAT PIC S9(9) COMP.
         29 SQL--MAXLENGTH PIC S9(9) COMP.
         29 SQL--SCALE PIC S9(9) COMP.
         29 SQL--PRECISION PIC S9(9) COMP.
         29 SQL--STTUS PIC S9(9) COMP.
         29 SQL--COUNT PIC S9(9) COMP.
         29 SQL--USERTYPE PIC S9(9) COMP.
         29 FILLER PIC S9(9) COMP. /* Second filler */
         29 SQL--LOCALE PIC S9(18) COMP. /* locale is */
                                  /* now eight bytes */
      19 SD-SQLDATA PIC S9(18) COMP.  /* SQLDATA is */
                                 /* now eight bytes */
      19 SD-SQLIND  PIC S9(4) COMP.
      19 FILLER PIC S9(4) COMP. /* Third filler */
      19 SD-SQLLEN  PIC S9(9) COMP.
      19 SD-SQLMORE PIC S9(18) COMP. /* SQLMORE is */
                                  /* now eight bytes */
```

SQLDA の 64 ビット ESQL/COBOL バージョンで、PIC 句に関連する要素は次のとおりです。

- S9(4) と定義されている要素 ― S9(4) は、ESQL/COBOL の 2 バイト長の smallint に相当します。64 ビット・アーキテクチャでは、メモリ・アドレスが 8 の倍数であることが必要となるため、S9(4) と定義されている要素は、単独では 64 ビットの要件を満たしていません。この要件を満たすには、S9(4) 要素を他の要素とグループ化するか、フィラーを使用して埋め込む必要があります。上記の 64 ビット・バージョンの SQLDA では、SD-SQLN と SD-SQLD を組み合わせても 4 バイト長にしかならないため、SD-SQLD の後に 4 バイト長のフィラーが追加されています。
- S9(9) と定義されている要素 ― S9(9) は、ESQL/COBOL の 4 バイト長の int に相当します。SQL-NMELEN と SQL-DATATYPE など、S9(9) 要素のペアは 64 ビットの整列要件を満たします。
- S9(18) と定義されている要素 ― S9(18) は、ESQL/COBOL の 8 バイト長のポインタまたは long に相当します。S9(18) と定義されている要素は、64 ビットのデータ整列要件を満たします。
- FILLER ― 上記の例では、データ構造体を埋め込み、8 バイト境界に配置するために、長さが異なる 3 つのフィラーが使用されています。

注意 フィラーを使用して SQLDA データ構造体を埋め込み、配置することはできますが、SQLDA データ構造体は修正しないでください。SQLDA 要素を追加または削除したり、要素の現在の定義を編集したりすることはできません。

ストアド・プロシージャのロード

プリコンパイラの -G フラグを使用してストアド・プロシージャを生成する場合は、isql を使用してストアド・プロシージャを Adaptive Server にロードしてから、プログラムを実行する必要があります。生成されたスクリプトを実行するための isql コマンドのフォーマットは、次のとおりです。

```
isql -Uuserid -Ppassword < program.sql
```

-U フラグには Adaptive Server にログインするためのユーザ ID を指定し、-P フラグにはパスワードを指定します。

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/COBOL のサンプル・プログラムの使用

Embedded SQL/COBOL プリコンパイラには、以下の項で説明するような、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・プログラムが提供されています。

注意 *$SYBASE/$SYBASE_OCS/sample/esqlcob* の内容を、元のファイルの整合性に影響を与えないでサンプル・プログラムを自由に使用できるような作業ディレクトリにコピーしてから、コンパイルして実行してください。

サンプル・プログラムの目的

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。

サンプル・プログラムは編集する必要があります。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。サンプル・プログラムを実行する方法の詳細については、*README* ファイルを参照してください。

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。実際の運用環境で使用できるレベルのプログラムでは、エラーや特殊なケースを処理するためのコードを追加する必要があります。

ロケーション

サンプル・プログラムは、*$SYBASE/$SYBASE_OCS/sample/esqlcob* にあります。

このディレクトリには次のファイルが含まれています。

- サンプル・プログラムのソース・コード

- サンプル・プログラムを構築するための *makefile*。*makefile* は、Server-Library アプリケーションの作成を開始するときに使用します。
- サンプル・プログラムの構築、実行、テストの方法について説明している *README* ファイル

sybopts.sh ファイルの所有者に、このファイルに対する execute パーミッションを設定します。

```
chmod u+x sybopts.sh
```

注意 サンプル・プログラムの結果を表示するために、[Enter] キーを押すことが必要な場合があります。

---

example1.pco サンプル・プログラム

example1.pco は、対話型クエリ・プログラムでのカーソルの使い方を示します。この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て、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 [Enter] キーが押されると、プログラムは終了します。

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ます。このプログラムは次のように動作します。

- authors 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著者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した後、データを Adaptive Server に送ります。

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、次のユーティリティ・プログラムのコマンド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。	92
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。	119
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。	130
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。	142
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。	148

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・ファイルからデータベース・テーブルにコピーします。このユーティリティは、*$SYBASE/$SYBASE_OCS/bin* にあります。

構文

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*]
[-M*LabelName LabelValue*] [-labeled]
[-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 *trusted.txt_file*]
[-X]
[-y *alternate_home_directory*]
[-Y]
[-z *language*]
[-Z *security_mechanism*]
[--colpasswd [[[db_name.[owner].]table_name.]
column_name [*password*]]]
[--keypasswd [[db_name.[owner].]*key_name* [*password*]]]

[--hide-vcc]
[--initstring “*TSQL_command*”]
[--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。そうでない場合は、データベース名を指定しなければなりません。

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。それ以外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しなければなりません。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。複数のパーティションの場合は、パーティション名のカンマ区切り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。複数のファイルを指定する場合は、ファイル名のカンマ区切りリストを使用します。複数のデータ・ファイルとパーティションの名前を入力する場合は、ファイルとパーティションの数が同じである必要があります。

-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 と -J を一緒に使用して、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。-a を使用するときに -J を省略できるのは、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場合だけです。

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。

大量のバルク・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なネットワーク・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・サイズは、バルク・コピー・インの場合にのみ適用されます。バルク・コピー・アウトには適用されません。bcp が *batchsize* に受け入れる最小数は 1 です。

注意 batchsize 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のみ役立ちます。-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、通常、領域の使用効率が低下します。

-c

char データ型をデフォルトとして使用してコピー・オペレーションを実行します。このオプションは各フィールドの入力を要求しません。デフォルトの記憶タイプとして char データ型を使用し、プレフィクスなしで、デフォルトのフィールド・ターミネータとして ¥t (タブ)、デフォルトのロー・ターミネータとして ¥n (復帰改行文字) 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、バルク・コピー・オペレーションの開始前に ciphertext オプションが有効化されます。

-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* 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 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* 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 は各ローに IDENTITY カラムのテンポラリの値 0 を割り当てます。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きに、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・ファイルからこの値を読み込み、この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 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取り除きます。-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 に対して使用するフォーマットを入力すると、bcp はフォーマット・ファイルとしてその形式を保存するかどうかを尋ねてきます。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。このオプションを使用しない場合は、フォーマット情報を対話的に入力する必要があります。

-F *firstrow*

コピーを開始する最初のローのロー番号を指定します (デフォルトは先頭のロー)。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 リリース・ディレクトリにある interfaces ファイル (*interfaces*) 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、『*ASE システム管理ガイド*』を参照してください。

-K *keytab_file*

(DCE セキュリティでのみ使用します)。-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 は、DCE dcecp ユーティリティを使用して作成します。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、bcp 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 *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。

-M *LabelName LabelValue*

(Secure SQL Server のみ) マルチレベル・ユーザがバルク・コピーのセッション・ラベルを設定できるようにします。*LabelName* の有効な値は次のとおりです。

- curread (現在の読み込みレベル) は、このセッション中に読み込むことができるデータの初期レベルです。curread は、curwrite よりも高いレベルにしてください。
- curwrite (現在の書き込みレベル) は、このセッション中に書き込むすべてのデータに 適用される初期 sensitivity レベルです。
- maxread (読み込みレベルの最大値) は、データを読み込むことができる最大レベルです。この値は、マルチレベル・ユーザとしてこのセッション中に curread に設定できる上限値です。maxread は、maxwrite よりも高いレベルにしてください。
- maxwrite (書き込み込みレベルの最大値) は、データを書き込むことができる最大レベルです。この値は、マルチレベル・ユーザとしてこのセッション中に curwrite に設定できる上限値です。maxwrite は、minwrite と curwrite よりも高いレベルにしてください。
- minwrite (書き込みレベルの最小値) は、データを書き込むことができる最小レベルです。この値は、マルチレベル・ユーザとしてこのセッション中に curwrite に設定できる下限値です。minwrite は、maxwrite と curwrite よりも低いレベルにしてください。

LabelValue は、システム上で使用される、人間の目で判読できるフォーマットで表現された実際のラベル値 (たとえば “Company Confidential Personnel”) です。

-labeled

(Secure SQL Server のみ) インポートしているデータの、すべてのレコードの最初のフィールドに、既にラベルがあることを示します。

エクスポートしているデータの場合、-labeled はすべてのローの sensitivity ラベルを最初のフィールドとしてコピー・アウトすることを示します。

-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 は各フィールドに対する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・フォーマットを使用して bcp を実行した場合、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 (たとえば、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) は、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、このオプションを使用します。

データをコピー・インするときに、-N オプションと -E オプション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。

-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 オプションまたは DSQUERY 環境変数で指定) と一致します。-R オプションは、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 データまたは image データの最大長をバイト単位で指定できます。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・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。

bcp のような SDK バイナリは、32 ビット版製品と 64 ビット版製品の両方で同じ名前を使用します。Adaptive Server、SDK、Open Server の 64 ビット版製品を他の Sybase 32 ビット版製品とともにインストールすると、32 ビット・バイナリが上書きされます。Adaptive Server 15.0.2 および SDK/Open Server 15.0 ESD #9 以降では、すべての 64 ビット UNIX プラットフォーム上で、64 ビット・バイナリが 32 ビット・バイナリで置き換えられています。32 ビット・バイナリは 64 ビット EBF に含まれるため、bcp の -v オプションは 64 ビット版製品の EBF 番号をチェックする有効な方法ではなくなっています。代わりに、UNIX の strings コマンドと grep コマンドを使用して、Open Client と Open Server の EBF 番号を確認します。

たとえば、*libsybct64.a* バイナリで EBF 番号を含む文字列を検索するには、次のように入力します。

```
strings -a libsybct64.a | grep Sybase
```

この場合、次のような文字列が返されます。

```
Sybase Client-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:04:17 2009
```

libsybsrv64.a バイナリで EBF 番号を含む文字列を検索するには、次のように入力します。

```
strings -a libsybsrv64.a | grep Sybase
```

この場合、次のような文字列が返されます。

```
Sybase Server-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:06:27 2009
```

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

代替の *trusted.txt* ファイルを指定します。

-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 がパスワードを暗号化するために使う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そのパスワードを確認します。

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 out の使用時に、サーバでの文字セット変換を無効にし、クライアント側で bcp を使用して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/config* 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。

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 UNIX 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 のデータを含む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。複数のファイルを並列にコピーする場合は、bcp_r (スレッド・バージョンの bcp ユーティリティ) を使用してください。たとえば、次の例では各オペレーションで許可する並列接続の最大数を 2 に設定します。

```
bcp_r --maxconn 2
```

このパラメータを指定しない場合、bcp はデフォルト値の 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。

例

例 1 この例では、-c オプションは、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 オプションは各フィールドをカンマで
終了し、-r row_terminator オプションは各行を改行文字で終了します。
bcp はパスワードだけを要求します。最後の "r" の前にある 1 つ目の円
記号は 2 つ目の円記号をエスケープするため、次のように出力される
円記号は 1 つだけです。

```
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 -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]:
```

次のように問い合わせてきます。

```
Do you want to save this format information in a
    file?[Y-n] y
Host filename [bcp.fmt]: pub_form
Starting copy...
3 rows copied.
 Clock time (ms.):total = 1 Avg = 0 (3000.00 rows per
sec.)
```

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 この例では、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 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 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 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ので、bcp の終了時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・ディレクトリを設定します。

```
bcp tempdb..T1 out T1.out -y/work/NewSybase -Uuser1
-Psecret -SMYSERVER
```

使用法

- bcp_r は、スレッド・バージョンの bcp です。Kerberos などのセキュリティ・サービスや LDAP などのディレクトリ・サービスを使用する場合は、bcp_r を使用してください。
- ファイルのコピー・イン、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リセットできます。
- *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 *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。
- *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。
- 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。
- 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・インする場合、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・アウトする場合は、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。変換エラーは次のようになっています。

  ```
  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 オプションを指定して 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。

  bcp の機能を以前のバージョンの機能に制限するには、*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。拒否されたローがない場合、破棄ファイルは作成されません。
- 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。
- -d 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* を指定しているが、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 コマンド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。しかし、bcp は、テーブルに定義された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たら、dump transaction を使用してログをトランケートできます。

- 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 database を発行すると、エラー・メッセージが表示され、代わりに dump transaction を使用するように指示されます。

警告！ select into/bulkcopy フラグをオフにする前に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 (対象のテーブルにインデックスやトリガがない場合)	実行可能 dump transaction の実行は不可	不可 Adaptive Server は低速 bcp を適用

	select into/ bulkcopy が on の場合	**select into/ bulkcopy が off の場合**
低速 bcp (1 つまたは複数のインデックスやトリガがある場合)	実行可能 dump transaction の実行は不可	実行可能 dump transaction は実行可能

- 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、まず drop index (またはインデックスの alter table) と drop trigger を使用して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) オプションまたは -c (文字フォーマット) オプションを使用して、データをコピー・インまたはコピー・アウトする場合、-P オプションでパスワードを指定していないと、bcp はパスワードだけを入力するよう要求します。-n、-c、または -f *formatfile* オプション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 では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 では char varbinary では 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) データ型で定義されるフィールドの場合 (char データ型とすべての固定長データ型) • 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 および 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。

プロンプト	デフォルト設定	可能な応答
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。
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、エントリに記憶領域の長さ分の埋め込みは行われません (char、nchar、binary データは、Adaptive Server から返されるときすでに、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、ファイルに入っている情報のプレフィクス長、ターミネータ、およびフィールド長の関係を示します。"P" は保管されたテーブルのプレフィックスを、"T" はターミネータを、2 つのダッシュ ("--") は追加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	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。
- ファイル記憶長は、通常はターミネータやプレフィクスを除く、カラムに転送されるデータの最大サイズを示します。
- 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、通貨単位の 1 万分の 1 の精度で money 値を保管します。しかし、money 値が文字フォーマットに変換される場合は、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章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。

- 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。

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。このユーティリティは、*$SYBASE/$SYBASE_OCS/bin* ディレクトリにあります。

注意 cpre64 と cpre_r64 は、64 ビット・アプリケーション用のプリコンパイラです。cpre64 プリコンパイラは、非リエントラント・プリコンパイラであり、cpre_r64 はリエントラント・バージョンです。これらのプリコンパイラは、Open Client と Open Server でサポートされるすべての 64 ビット・プラットフォームで使用できます。

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*]

[-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指定できます。たとえば、cpre -l と cpre /l は同じことを表します。

パラメータ

-C *compiler*
対象のホスト言語コンパイラの値を指定します。たとえば、ANSI C コンパイラの場合は "ANSI_C" を指定します。

-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 も指定すると、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。現在のところ、プリコンパイラは SQL89 または SQL92E を調べます。

-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。*isql_file_name* はオプションです。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 拡張子を付加した入力ファイル名、または既存の入力ファイル名の拡張子をこの拡張子で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、-T tag_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
フェイ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に複数のディレクトリを探します。このオプションを指定しないときのデフォルトは、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文字セットを使用していると解釈します。

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 は次の順序で情報を検索します。

1　LC_ALL

2　LANG

LC_ALL が定義されている場合、CS-Library はこの値をロケール名として使用します。LC_ALL が定義されておらず、LANG が定義されている場合は、CS-Library はその値をロケール名として使用します。これらのロケール値がど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。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* ファイル内からロケール名を探して、そのロケール名に対応す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。

- none (デフォルト)
- 構文
- semantic

SYNTAX または SEMANTIC を指定する場合は、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 つ以上のリスティング・ファイルを生成します。*listing_file_name* (オプションの引数) は、各行に番号が付けられ、該当するエラー・メッセージが含まれる入力ファイルです。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* 拡張子を付加した入力ファイル名、または既存の入力ファイル名の拡張子をこの拡張子で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インタフェース・ファイルの名前、*interfaces* 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ないため、デフォルトのターゲット名が割り当てられます。デフォルトのターゲット名は、*.cbl* 拡張子を付加した入力ファイル名、または既存の入力ファイル名の拡張子をこの拡張子で置き換えた入力ファイル名になります。

-P *password* (-Uuser_id オプションとともに使用)
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 の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 オプションとともに使用)
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 (最大 3 文字) を指定します。

たとえば、-T dbg 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。-Ppassword 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 の各オプション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順序に従って決定します。

1 ロケール名を探します。CS-Library は、情報を LC_ALL でまず検索し、次に LANG で検索します。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 *$SYBASE/locales* にある *locales.dat* ファイルのロケール名を検索して、関連付けられている言語と文字セットを決定します。

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 が有効であるかのように動作します。これは ANSI 準拠の動作です。カーソルとトランザクションの詳細については、『*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指定をしないかぎり、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 11.1 バージョンと 10.x バージョンで次のような相違点があります。

- 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のときには役立ちますが、最終的にアプリケーションを配布するときにはオフにしてください。Sybase リリース・ディレクトリの *$SYBASE/$SYBASE_OCS/devlib* ディレクトリ内にあるライブラリと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
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。このオプションを使用すると、入力パラメータの付い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改善されます。ただし、格納領域の使用量が増加して、各コマンドの初回の応答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*』の CS_EXTERNAL_CONFIG プロパティの説明と、『*Open Client Embedded SQL/C* プログラマーズ・ガイド』の 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、適用される規則に従っていれば、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、次のどちらのフォーマットでも使用できます。

  -T dbg
  または
  -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。ただし、-O *target_file_name* オプションは使用できません。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] 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・プロンプトで実行してください。

1 構文チェックとデバッグを行うために、-c、-Ddatabase_name、-Ppassword、-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 -Gisql_file_name
```

3　プログラム上でテストを実行します。

4　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 -Gisql_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・ラインでサーバを指定するには、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からサーバ名を省略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 UNIX 版』を参照してください。

cpre のデフォルト

表 A-6 に、cpre ユーティリティと cobpre ユーティリティ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* 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 *interfaces* 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で置き換えた) 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。このユーティリティは、*$SYBASE/$SYBASE_OCS/bin* にあります。

注意 cobpre64 と cobpre_r64 は、64 ビット・アプリケーション用のプリコンパイラです。cobpre64 プリコンパイラは、非リエントラント・プリコンパイラであり、cobpre_r64 はリエントラント・バージョンです。

構文

cobpre
[-C*compiler*]
[-D*database*_name]
[-F*ips_level*]
[-G[*isql_file_name*]]
[-I*include_path_name*]
[-J*charset_locale_name*]
[-K*syntax_level*]
[-L[*listing_file_name*]]
[-M]
[-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 (/) またはハイフン (-) のどちらを使用してもオプションを指定できます。たとえば、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 も指定すると、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。現在のところ、プリコンパイラは SQL89 または SQL92E を調べます。

-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。isql_file_name はオプションです。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* 拡張子を付加した入力ファイル名、または既存の入力ファイル名の拡張子をこの拡張子で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、-T tag_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。このオプションを指定しないときのデフォルトは、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文字セットを使用していると解釈します。

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 は次の順序で情報を検索します。

1　LC_ALL

2　LANG

LC_ALL が定義されている場合、CS-Library はこの値をロケール名として使用します。LC_ALL が定義されておらず、LANG が定義されている場合は、CS-Library はその値をロケール名として使用します。これらのロケール値がど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。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。

- none (デフォルト)
- 構文
- semantic

syntax または semantic を指定する場合は、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 つ以上のリスティング・ファイルを生成します。*listing_file_name* (オプションの引数) は、各行に番号が付けられ、該当するエラー・メッセージが含まれる入力ファイルです。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* 拡張子を付加した入力ファイル名、または既存の入力ファイル名の拡張子をこの拡張子で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 (*interfaces*) 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ないため、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。デフォルトのターゲット・ファイル名は、*.cbl* 拡張子を付加した入力ファイル名、または既存の入力ファイル名の拡張子をこの拡張子で置き換えた入力ファイル名になります。

-P *password* (-Uuser_id オプションとともに使用)

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 の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 オプションとともに使用)
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 (最大 3 文字) を指定します。

たとえば、-T dbg 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 の各オプション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_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順序に従って決定します。

1 ロケール名を探します。CS-Library は、情報を LC_ALL でまず検索し、次に LANG で検索します。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 が有効であるかのように動作します。これは ANSI 準拠の動作です。カーソルとトランザクションの詳細については、『*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指定をしないかぎり、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 11.1 バージョンと 10.x バージョンで次のような相違点があります。

- 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のときには役立ちますが、最終的にアプリケーションを配布するときにはオフにしてください。Sybase リリース・ディレクトリの *%SYBASE%¥%SYBASE_OCS¥devlib* ディレクトリにあるライブラリおよび DLL と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』の CS_EXTERNAL_CONFIG プロパティの説明と、『*Open Client Embedded SQL/C* プログラマーズ・ガイド』の 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、適用される規則に従っていれば、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 -K SEMANTIC -Uuser_id -Ppassword -Sserver_name ¥
       -Dpubs2 example1.pco
```

使用法

- cobpre| コマンドのデフォルトは、ANSI 標準の動作に設定されます。
- 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の拡張子で置き換えた) 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、次のどちらのフォーマットでも使用できます。

  -T dbg
  または
  -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。ただし、-O *target_file_name* オプションは使用できません。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・プロンプトで実行してください。

1 構文チェックとデバッグを行うために、-Ddatabase_name、-Ppassword、-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 -Gisql_file_name
```

3 プログラム上でテストを実行します。

4　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 -Gisql_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・ラインでサーバを指定するには、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からサーバ名を省略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 UNIX 版』を参照してください。

cobpre | cpre のデフォルト

cpre ユーティリティおよび cobpre ユーティリティのオプションとデフォルトのリストについては、表 A-6 (130 ページ) 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]]
    [-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 が実行されるマシンの文字セットと異なる文字セットの端末から 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。-a を使用するときに -J を省略できるのは、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 リリース・ディレクトリにある interfaces ファイル (*interfaces*) 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している文字セットと同じであるとはかぎりません。

-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 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・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。

defncopy のような SDK バイナリは、32 ビット版製品と 64 ビット版製品の両方で同じ名前を使用します。Adaptive Server、SDK、Open Server の 64 ビット版製品を他の Sybase 32 ビット版製品とともにインストールすると、32 ビット・バイナリが上書きされます。Adaptive Server 15.0.2 および SDK/Open Server 15.0 ESD #9 以降では、すべての 64 ビット UNIX プラットフォーム上で、64 ビット・バイナリが 32 ビット・バイナリで置き換えられています。32 ビット・バイナリは 64 ビット EBF に含まれるため、defncopy の -v オプションは 64 ビット版製品の EBF 番号をチェックする有効な方法ではなくなっています。代わりに、UNIX の strings コマンドと grep コマンドを使用して、Open Client と Open Server の EBF 番号を確認します。

たとえば、*libsybct64.a* バイナリで EBF 番号を含む文字列を検索するには、次のように入力します。

```
strings -a libsybct64.a | grep Sybase
```

この場合、次のような文字列が返されます。

```
Sybase Client-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:04:17 2009
```

libsybsrv64.a バイナリで EBF 番号を含む文字列を検索するには、次のように入力します。

```
strings -a libsybsrv64.a | grep Sybase
```

この場合、次のような文字列が返されます。

```
Sybase Server-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:06:27 2009
```

-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 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。

-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/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 UNIX 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、*owner* 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、指定したファイルからすべての定義をコピー・インします。

  定義をコピー・アウトするには、sysobjects テーブルおよび syscomments テーブルに対する select パーミッションが必要です。オブジェクト自体のパーミッションは必要ありません。

- コピー・インするオブジェクトのタイプについては、適切な create パーミッションを持つ必要があります。コピー・インされたオブジェクトはコピーするユーザが所有します。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！ 100 文字を超える長いコメントが create 文の前にあると、defncopy が失敗することがあります。

---

isql

説明

Adaptive Server の Interactive 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] [-W] [-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*]
[-M*LabelName LabelValue*]
[-o *outputfile*]
[-P *password*]
[-R *remote_server_principal*]
[-s *col_separator*]
[-S *server_name*]
[-t *timeout*]
[-U *username*]
[--URP *remotepassword*
[-V [*security_options*]]
[-w *column_width*]
[-x *trusted.txt_file*]
[-y *sybase_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。-a を使用するときに -J を省略できるのは、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 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 (vi 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 フラガを有効にします。-F パラメータを指定した場合、サーバは非標準 SQL 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のではありません。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 *inputfilename*

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・ターミネータを含むようにしてください (デフォルトでは “go”)。

- 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 リリース・ディレクトリにある interfaces ファイル (*interfaces*) 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。文字セットおよび関連するフラグの詳細については、『*ASE システム管理ガイド*』を参照してください。

-K *keytab_file*

DCE セキュリティでのみ使用します。*keytab_file* は、-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、dcecp ユーティリティを使用して作成します。詳細については、DCE のマニュアルを参照してください。

-K を指定しない場合、bcp 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。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。

-M *LabelName LabelValue*

(Secure SQL Server のみ) マルチレベル・ユーザがバルク・コピーのセッション・ラベルを設定できるようにします。*LabelName* の有効な値は次のとおりです。

- curread (現在の読み込みレベル) は、このセッション中に読み込むことができるデータの初期レベルです。curread は、curwrite よりも高いレベルにしてください。
- curwrite (現在の書き込みレベル) は、このセッション中に書き込むすべてのデータに 適用される初期 sensitivity レベルです。
- maxread (読み込みレベルの最大値) は、データを読み込むことができる最大レベルです。この値は、マルチレベル・ユーザとしてこのセッション中に curread に設定できる上限値です。maxread は、maxwrite よりも高いレベルにしてください。
- maxwrite (書き込み込みレベルの最大値) は、データを書き込むことができる最大レベルです。この値は、マルチレベル・ユーザとしてこのセッション中に curwrite に設定できる上限値です。maxwrite は、minwrite と curwrite よりも高いレベルにしてください。
- minwrite (書き込みレベルの最小値) は、データを書き込むことができる最小レベルです。この値は、マルチレベル・ユーザとしてこのセッション中に curwrite に設定できる下限値です。minwrite は、maxwrite と curwrite よりも低いレベルにしてください。

LabelValue は、システム上で使用される、人間の目で判読できるフォーマットで表現された実際のラベル値 (たとえば “Company Confidential Personnel”) です。

-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 *output_filename*

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*

現在の Adaptive Server 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機能を提供します。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*

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 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 *colseparator*

カラム・セパレータ文字を再設定します。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*

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、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、isql 内から発行されたコマンドに影響するもので、接続時間には影響しません。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。

--URP *remotepassword*

Adaptive Server にアクセスするクライアントに対してユニバーサル・リモート・パスワード (*remotepassword*) の設定を有効にします。

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

isql のような SDK バイナリは、32 ビット版製品と 64 ビット版製品の両方で同じ名前を使用します。Adaptive Server、SDK、Open Server の 64 ビット版製品を他の Sybase 32 ビット版製品とともにインストールすると、32 ビット・バイナリが上書きされます。Adaptive Server 15.0.2 および SDK/Open Server 15.0 ESD #9 以降では、すべての 64 ビット UNIX プラットフォーム上で、64 ビット・バイナリが 32 ビット・バイナリで置き換えられています。32 ビット・バイナリは 64 ビット EBF に含まれるため、isql の -v オプションは 64 ビット版製品の EBF 番号をチェックする有効な方法ではなくなっています。代わりに、UNIX の strings コマンドと grep コマンドを使用して、Open Client と Open Server の EBF 番号を確認します。

たとえば、*libsybct64.a* バイナリで EBF 番号を含む文字列を検索するには、次のように入力します。

```
strings -a libsybct64.a | grep Sybase
```

この場合、次のような文字列が返されます。

```
Sybase Client-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:04:17 2009
```

libsybsrv64.a バイナリで EBF 番号を含む文字列を検索するには、次のように入力します。

```
strings -a libsybsrv64.a | grep Sybase
```

この場合、次のような文字列が返されます。

```
Sybase Server-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:06:27 2009
```

-w *columnwidth*

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。

-x *trusted.txt_file*

代替の *trusted.txt* ファイルを指定します。

-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 がパスワードを暗号化するために使う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そのパスワードを確認します。

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 *sybase_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/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 UNIX 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/config/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・モードでは無視されます。

「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」(168 ページ) を参照してください。

--help
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が表示されます。

--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 ー 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* ー 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* ー 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 *$HOME/.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」(169 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。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 クエリを編集できるテキスト・エディタを開きます。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 この retserverror の例では、重大度が 16 のサーバ・エラーが発生すると、isql は呼び出し元シェルに 2 を返し、stderr に "Msg 207" を出力して終了します。

```
guest> 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'.


guest> echo $?
2
guest> cat isql.stderr
Msg     207
guest>
```

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。

```
guest> isql --help

使用方法 : isql [option1] [option2] ... where [options] are...
-b                    テーブル・ヘッダの出力表示を無効にします。
-e                    入力内容をエコーします。
-F                    FIPS フラガを有効にします。
-p                    パフォーマンスの統計値を出力します。
-n                    -e とともに使用した場合、行番号とプロンプト記号を
                      削除します。
-v                    バージョン番号と版権メッセージを表示します。
-W                    接続再試行に対する拡張パスワード暗号化を
                      オフにします。
-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。
-Y                    Adaptive Server に、連鎖トランザクションを使用するよう伝えます。
-Q                    HAFAILOVER プロパティを有効にします。
-a display_charset -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。
                      -a を使用するときに -J を省略できるのは、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場合だけです。
-A packet_size        この isql 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。
-c cmdend             コマンド・ターミネータを変更します。
-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 editor             デフォルトのエディタである vi 以外のエディタを指定します。
-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。
-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。
-I interfaces_file interfaces ファイルの名前とロケーションを指定します。
-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。
```

-K *keytab_file*　　DCE での認証に使用する keytab ファイルのパスを指定します。

-l *login_timeout*　　ログインの試行にサーバが応答するまで待機する秒数を指定します。

-m *errorlevel*　　エラー・メッセージの表示をカスタマイズします。

-M *labelname labelvalue*

セキュリティ・ラベルに使用します。詳細については、CS_SEC_NEGOTIATE を参照してください。

-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。

-P *password*　　Adaptive Server のパスワードを指定します。

-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。

-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。

-S *server_name*　　接続する Adaptive Server の名前を指定します。

-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。

-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。有効な [*security_options*] 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 — 接続の確立に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。
- d — クレデンシャル委任を要求し、クライアントのクレデンシャルを転送する。

-w *column_width*　　出力画面の幅を設定します。

-y *sybase_directory*

Sybase ホーム・ディレクトリを別のロケーションに設定します。

-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を設定します。

-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。

-x *trusted.txt_file*　　代替の trusted.txt ファイルのロケーションを指定します。

--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。

--conceal [*wildcard*]

ISQL セッションでの入力を難読化します。オプションのワイルドカードはプロンプトとして使用されます。

例 7 -y オプションを使用して、代替の Sybase ホーム・ディレクトリを設定します。

```
isql -y/work/NewSybase -Uuser1 -Psecret -SMYSERVER
```

例 8 --conceal のこの例では、入力したパスワードを表示せずにパスワードを変更します。次の例では、プロンプトのラベルとして “old” と “new” を使用します。

```
$ 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 --conceal のこの例では、入力したパスワードを表示せずにパスワードを変更します。次の例では、プロンプトのラベルとしてデフォルト・ワイルドカードを使用します。

```
$ 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。--conceal のこの例では、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。

```
$ 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
  ```

- T-SQL コマンドの出力を新しいファイルにリダイレクトするか、ファイルが既に存在する場合はそのファイルを上書きする場合：

  ```
  >
  ```

- T-SQL コマンドの出力を新しいファイルにリダイレクトするか、ファイルが既に存在する場合はそのファイルに追加する場合：

  ```
  >>
  ```

- T-SQL コマンドの出力を isql セッション内から外部アプリケーションにパイプする場合：

  ```
  |
  ```

- isql は、Client-Library を使用して構築されています。isql は、非スレッド Client-Library を使用して構築されています。
- isql_r は、スレッド・バージョンの isql です。Kerberos などのセキュリティ・サービスや LDAP などのディレクトリ・サービスを使用する場合は、isql_r を使用してください。
- エラー・メッセージのフォーマットは、isql の前のバージョンと異なっています。これらのメッセージの内容に基づいたルーチンを実行するスクリプトは、修正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 (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、あるコマンドを 100 回実行する場合は、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 vi です。

  たとえば、EDITOR 環境変数を *emacs* に設定している場合は、行の最初の単語に *emacs* を使用して、isql から emacs を呼び出します。

- 行頭に 2 つの感嘆符 (!!) を付けてコマンドを入力すると、オペレーティング・システム・コマンドを実行できます。

- 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。このエントリは isql を使用して保留中の点を廃棄します。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 -U alma -P****** < input_file
  ```

  このファイルには、コマンド・ターミネータが含まれていなければなりません。結果は端末に表示されます。次のようにして、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ことができます。

  ```
  isql -U alma -P****** < input_file > output_file
  ```

- isql フラグでは、大文字と小文字が区別されます。

- 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、編集を終わったあとにターミネータを対話的に入力してください。
- 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 を対話的に使用するときは、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 文にコメントを付けることができます。コメントは、次の例に示すように “/*” 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にする場合は、コマンドが行の先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
  */
  ```

  次のような使い方はできません。

  ```
  /*
  go
  */
  ```

- isql では、ロケール環境に関係なく、日付フォーマットの順序が月、日、年の順で定義されています (mm dd yyy hh:mm AM または PM)。このデフォルトの順序を変更するには、convert 関数を使用します。

isql 内で使用できるその他のコマンド

表 A-7: isql セッション・コマンド

コマンド	説明
>	コマンド出力をファイルにリダイレクトする。ファイルが存在する場合は、そのファイルが上書きされる。
>>	コマンド出力をファイルにリダイレクトする。ファイルが既に存在する場合は、そのファイルに出力内容が追加される。
|	コマンドの出力を外部アプリケーションにパイプする。
reset	クエリ・バッファをクリアします。
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・コマンドを実行します。
: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。

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。プロンプトのラベルを 1 単語より多く指定した場合、最初の単語より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た場合だけです。

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 ー 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。

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

付 録　B

環境変数

この付録では、Sybase アプリケーションの正しいコンパイルおよび動作のために必要な環境変数の値について説明します。設定しなければならない環境変数は、使用するアプリケーションによって異なります。また、次のような環境変数が含まれます。

- SYBASE － Sybase インストール・ディレクトリのパスを設定します。
- SYBASE_OCS － Open Client/Server のバージョン番号を含むサブディレクトリを設定します。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 － Adaptive Server または Open Server の名前を設定します。
- DSLISTEN － Open Server の名前を設定します。
- SYBPLATFORM － 使用するプラットフォームと、リエントラント・ライブラリを使用するかどうかによって適切な値を設定します。この環境変数の適切な設定値については、表 B-1 を参照してください。
- 表 B-1 にリストされているプラットフォーム固有のライブラリ・パス変数を *$SYBASE/$SYBASE_OCS/lib* に設定して、共有 (動的) ライブラリにリンクしているプログラムを実行してください。debug モードでプログラムを実行するときは、プラットフォーム固有のライブラリ・パス変数を *$SYBASE/$SYBASE_OCS/devlib* に設定してください。

  ESQL/COBOL アプリケーションの場合は、*$COBDIR/coblib* ディレクトリのロケーションをインクルードします。

表 B-1: SYBPLATFORM とライブラリ・パス

プラットフォーム	SYBPLATFORM の設定	プラットフォーム固有のライブラリ・パス変数
HP HP-UX PA-RISC 32 ビット版 HP HP-UX PA-RISC 32 ビット版 (ネイティブ・スレッドを使用)	hpux nthread_hpux	SHLIB_PATH
HP HP-UX PA-RISC 64 ビット版 HP HP-UX PA-RISC 64 ビット版 (ネイティブ・スレッドを使用)	hpux64 nthread_hpux64	LD_LIBRARY_PATH
HP HP-UX Itanium 32 ビット版 HP HP-UX Itanium 32 ビット版 (ネイティブ・スレッドを使用)	hpia nthread_hpia	SHLIB_PATH
HP HP-UX Itanium 64 ビット版 HP HP-UX Itanium 64 ビット版 (ネイティブ・スレッドを使用)	hpia64 nthread_hpia64	LD_LIBRARY_PATH
IBM AIX RS/6000 32 ビット版 IBM AIX RS/6000 32 ビット版 (ネイティブ・スレッドを使用)	rs6000 nthread_rs6000	LIBPATH
IBM AIX RS/6000 64 ビット版 IBM AIX RS/6000 64 ビット版 (ネイティブ・スレッドを使用)	rs600064 nthread_rs600064	LIBPATH
Linux x86 32 ビット版 Linux x86 32 ビット版 (ネイティブ・スレッドを使用)	linux nthread_linux	LD_LIBRARY_PATH
Linux POWER 32 ビット版 Linux POWER 32 ビット版 (ネイティブ・スレッドを使用)	ibmplinux nthread_ibmplinux	LD_LIBRARY_PATH

プラットフォーム	SYBPLATFORM の設定	プラットフォーム固有のライブラリ・パス変数
Linux POWER 64 ビット版	ibmplinux64	LD_LIBRARY_PATH
Linux POWER 64 ビット版 (ネイティブ・スレッドを使用)	nthread_ibmplinux64	
Linux x86-64 64 ビット版	linux64 または linuxamd64	LD_LIBRARY_PATH
Linux x86-64 64 ビット版 (ネイティブ・スレッドを使用)	nthread_linux64 または nthread_linuxamd64	
Solaris SPARC 32 ビット版	sun_svr4	LD_LIBRARY_PATH
Solaris SPARC 32 ビット版 (ネイティブ・スレッドを使用)	nthread_sun_svr4	
Solaris SPARC 64 ビット版	sun_svr464	LD_LIBRARY_PATH_64
Solaris SPARC 64 ビット版 (ネイティブ・スレッドを使用)	nthread_sun_svr464	
Solaris x86-64 32 ビット版	sunx86	LD_LIBRARY_PATH
Solaris x86-64 32 ビット版 (ネイティブ・スレッドを使用)	nthread_sunx86	
Solaris x86-64 64 ビット版	sunx8664	LD_LIBRARY_PATH_64
Solaris x86-64 64 ビット版 (ネイティブ・スレッドを使用)	nthread_sunx8664	

Embedded SQL/COBOL アプリケーションの場合は、上記の環境変数に加えて、次の環境変数を設定します。

- COBDIR － COBOL コンパイラのパスを設定する。
- PATH － $COBDIR/bin を追加する。

付 録　C

ユーティリティ・メッセージ

この付録では、 bcp 、 defncopy 、 isql の各ユーティリティのエラー、情報、警告のメッセージについて説明します。

bcp メッセージ

メッセージ 1 ：メモリ割り当ての失敗

メッセージ・タイプ	エラー
記号定数	BERRNOMEM
メッセージ	致命的なエラー：メモリの割り当てに失敗しました。
原因	*start* 引数が無効です。
対処法	*start* 引数が言語または RPC コマンドの長さより短いことを確認します。

メッセージ 5 ：入力ファイルを開けない

メッセージ・タイプ	エラー
記号定数	BERRNOINFILE
メッセージ	入力ファイル `'%1!'` をオープンできません。
原因	bcp が入力用のデータ・ファイルを開くことができません。
対処法	指定されたファイルを確認します。

メッセージ 6 : 出力ファイルを開けない

メッセージ・タイプ エラー

記号定数 BERRNOOUTFILE

メッセージ 出力ファイル '%1!' をオープンできません。

原因 bcp が出力用のデータ・ファイルを開くことができません。

対処法 指定されたファイルを確認します。

メッセージ 7 : 不正な引数

メッセージ・タイプ エラー

記号定数 BERRBADARG

メッセージ bcp : 不明なパラメータ '%1!'。

原因 不明なパラメータが発行されました。

対処法 不明なパラメータを修正し、再発行します。

メッセージ 8 : 無効な最初のロー

メッセージ・タイプ エラー

記号定数 BERRFIRSTROW

メッセージ コピーする最初のローの設定に '%1!' フラグを使用するとき、ロー番号は 1 以上でなければなりません。

原因 指定された最初のローが無効です。

対処法 ロー番号を修正します。

メッセージ 9 : 無効なロー

メッセージ・タイプ エラー

記号定数 BERRFLROW

メッセージ コピーする最初のローと最後のローの設定に '%1!' および '%2!' フラグを使用するとき、最初のロー番号は最後のロー番号よりも小さくなければなりません。

原因　指定された最初のローまたは最後のローが無効です。

対処法　最初のロー番号が最後のロー番号より小さくなるように、ローの範囲を修正します。

メッセージ 10 ：無効な最後のロー

メッセージ・タイプ　エラー

記号定数　BERRLASTROW

メッセージ　コピーする最後のローの設定に `'%1!'` フラグを使用するとき、ロー番号は `1` 以上でなければなりません。

原因　ロー番号を修正します。

対処法　*start* 引数が言語または RPC コマンドの長さより短いことを確認します。

メッセージ 11 ：無効な方向

メッセージ・タイプ　エラー

記号定数　BERRBADDIR

メッセージ　コピーする方向は「イン」か「アウト」のどちらかです。

原因　指定された方向が無効です。

対処法　方向を修正します。

メッセージ 12 ：無効な整数

メッセージ・タイプ　エラー

記号定数　BERRBADINTARG

メッセージ　`'%1!'` フラグのあとに整数を入力してください。`'%2!'` は整数ではありません。

原因　指定された引数が整数ではありません。

対処法　引数を修正します。

メッセージ 13 : フラグの重複

メッセージ・タイプ 警告

記号定数 BERRDUPARGS

メッセージ 警告 : '%1!' フラグが 2 個以上あります。新しいフラグの値は既存の値よりも大きいものになります。

原因 重複する引数が指定されました。

対処法 どちらかの引数を削除します。

メッセージ 14 : 引数の上書き

メッセージ・タイプ 警告

記号定数 BERROVERRIDE

メッセージ 警告 : '%1!' は '%2!' を上書きします。

原因 2 つの引数がお互いを上書きしています。

対処法 必要に応じてどちらかの引数を削除します。

メッセージ 15 : 無効なプレフィクス長

メッセージ・タイプ エラー

記号定数 BERRBADPREFXLEN

メッセージ プレフィクス長が無効です。正しいプレフィクス長は、0、1、2 または 4 です。

原因 プレフィクス長が無効です。

対処法 有効なプレフィクス長を指定します。

メッセージ 21 : 再試行

メッセージ・タイプ 情報

記号定数 BSTRTRY

メッセージ もう一度やり直してください

原因 致命的でないエラーが発生しました。

対処法 再度操作を実行します。

メッセージ 23 : 開始メッセージ

メッセージ・タイプ　　情報

記号定数　　BSTRSTART

メッセージ　　コピーしています ...

メッセージ 24 : N ローのコピー

メッセージ・タイプ　　情報

記号定数　　BSTRROW

メッセージ　　%1! ローをコピーしました。

メッセージ 25 : 合計時間

メッセージ・タイプ　　情報

記号定数　　BSTRTIME

メッセージ　　クロック・タイム (ミリ秒) : 合計 = %1!

メッセージ 26 : ファイルの保存

メッセージ・タイプ　　情報

記号定数　　BSTRSAVE

メッセージ　　このフォーマット情報をファイルに保存しますか? [Y/n]

メッセージ 27 : ホスト・ファイル

メッセージ・タイプ　　情報

記号定数　　BSTRHOST

メッセージ　　ホストファイル名 [%1!] :

メッセージ 28 : 無効なカラム・タイプ

メッセージ・タイプ エラー

記号定数 BERRCOLTYPE

メッセージ カラムの型が無効です。次の型が有効です :

原因 指定されたカラム・タイプが無効です。

対処法 有効なカラム・タイプを指定します。

メッセージ 29 : 無効なカラム・タイプ

メッセージ・タイプ 情報

記号定数 BERRDSCOL

メッセージ `<cr>: DataServer` カラムと同じ型です。

原因 指定されたカラム・タイプが無効です。

対処法 有効なカラム・タイプを指定します。

メッセージ 30 : 平均時間

メッセージ・タイプ 情報

記号定数 BSTRAVG

メッセージ 平均 = `%1!` (`%2!` ロー / 秒)

ローあたりの平均処理時間を示しています。

メッセージ 31 : コピーの失敗

メッセージ・タイプ エラー

記号定数 BERRCOPY

メッセージ `%1!` の `bcp` によるコピーに失敗しました。

原因 コピー中にエラーが発生しました。

対処法 再度コピーを実行します。

メッセージ 32 : 部分的なコピーの失敗

メッセージ・タイプ　エラー

記号定数　BERRPCOPY

メッセージ　%1! の bcp によるコピーに一部失敗しました。

原因　いくつかのローがコピーされませんでした。

対処法　特定のローに対して再度コピーを実行します。

メッセージ 33 : 無効な精度

メッセージ・タイプ　エラー

記号定数　BERRBADPRECISION

メッセージ　精度が無効です。精度は %1! から %2! のあいだでなければなりません。

原因　指定された精度が無効です。

対処法　有効な精度を指定します。

メッセージ 34 : 無効な位取り

メッセージ・タイプ　エラー

記号定数　BERRBADSCALE

メッセージ　位取りが無効です。位取りは %1! から %2! のあいだで、精度よりも小さいか等しい値に設定してください

原因　指定された位取りが無効です。

対処法　有効な位取りを指定します。

メッセージ 35 : 予期しない結果タイプ

メッセージ・タイプ　エラー

記号定数　BERRBADTYPE

メッセージ　予期しない結果タイプが返されました。

原因　サーバーから不適切な結果タイプが返されました。

メッセージ 36 : 予期しない結果

メッセージ・タイプ エラー

記号定数 BERRBADRESULT

メッセージ 予期しない結果が返されました。

原因 サーバーから不適切な結果が返されました。

メッセージ 37 : 書き込みエラー

メッセージ・タイプ エラー

記号定数 BERRWRITEERR

メッセージ `BCP` ファイル `(%1!)` の書き込みエラーです。

原因 bcp ファイルへの書き込みでエラーが発生しました。

対処法 指定されたファイルを確認します。

メッセージ 39 : 無効なロー

メッセージ・タイプ エラー

記号定数 BERRNOTENOUGHROWS

メッセージ 指定された最初のローがテーブル内のロー数を越えています。

原因 指定された最初のローの番号が、テーブル内のロー数を超えています。

対処法 最初のローとして有効な値を指定します。

メッセージ 40 : ローの転送エラー

メッセージ・タイプ エラー

記号定数 BERRXFERMULT

メッセージ `blk_rowxfer_mult` が予期しないリターン・コードを返しました。

原因 blk_rowxfer_mult ルーチンからのリターン・コードが予期しないものでした。

メッセージ 41 : データ型が無効

メッセージ・タイプ　エラー

記号定数　BERRDATATYPE

メッセージ　'%1!' は認識できないデータ型です。

原因　コピー時に無効なデータ型が検出されました。

メッセージ 42 : 入力ファイルの読み込みエラー

メッセージ・タイプ　エラー

記号定数　BERRIOERR

メッセージ　bcp 入力ファイルの読み込み中に I/O エラーが発生しました。

原因　bcp 入力ファイルの読み込みでエラーが発生しました。

対処法　指定されたファイルを確認します。

メッセージ 43 : エラー・ファイルの書き込みエラー

メッセージ・タイプ　エラー

記号定数　BERRWEF

メッセージ　bcp 入力ファイルへの書き込み中に I/O エラーが発生しました。

原因　bcp エラー・ファイルへの書き込みでエラーが発生しました。

対処法　指定されたファイルを確認します。

メッセージ 44 : エラー・ファイルを開けない

メッセージ・タイプ　エラー

記号定数　BERRUOE

メッセージ　エラー・ファイルをオープンできません。

原因　bcp がエラー・ファイルを開くことができません。

対処法　指定されたファイルを確認します。

メッセージ 45 : 予期しないファイル終了記号

メッセージ・タイプ エラー

記号定数 BERREOF

メッセージ BCP データ・ファイル内で予期しない EOF を検出しました。

原因 bcp データ・ファイル内に予期しないファイル終了記号がありました。

対処法 指定されたファイルの内容を確認します。

メッセージ 46 : 負の長さプレフィクス

メッセージ・タイプ エラー

記号定数 BERRCNL

メッセージ BCP データ・ファイル内で値が負の長さプレフィクスを検出しました。

原因 bcp データ・ファイル内に負の長さプレフィクスがありました。

対処法 bcp データ・ファイル内で有効な長さプレフィクスを指定します。

メッセージ 48 : 指定されたロー数を読み込めない

メッセージ・タイプ エラー

記号定数 BERRSHORTFILE

メッセージ BCP ホスト・ファイル '%1!' には %2! ローしかありません。要求された数のローを読み込むことはできません。

原因 bcp ホスト・ファイル内のロー数は、読み込みが要求されたロー数よりも少ない値です。

対処法 読み込むロー数には、 bcp ホスト・ファイル内のロー数以下の値を指定します。

メッセージ 49 : 長さプレフィクスまたはターミネータが必要

メッセージ・タイプ　エラー

記号定数　BERRVDPT

メッセージ　バルク・コピーを実行するには、すべての可変長データに長さプレフィクスまたはターミネータを指定してください。

原因　バルク・コピーを実行するには、すべての可変長データに長さプレフィクスまたはターミネータが指定されている必要があります。

対処法　長さプレフィクスまたはターミネータを指定します。

メッセージ 50 : text および image データのトランケート

メッセージ・タイプ　エラー

記号定数　BERRTRUNDATA

メッセージ　`Text/image` フィールドが最大値を上回っています。データはトランケートされます。

原因　text または image データが最大サイズを超えています。最大サイズを超えるデータがトランケートされました。

メッセージ 51 : 最大エラー数の超過

メッセージ・タイプ　エラー

記号定数　BERRMAXERROR

メッセージ　`BCP` 処理中のエラーの合計数が、エラーの最大許容数 (`%1!`) を超過しました。 `BCP` は停止しました。

原因　bcp 処理で、エラーの最大許容数を超えました。

メッセージ 52 : 破棄ファイルを開けない

メッセージ・タイプ　エラー

記号定数　BERRUOD

メッセージ　破棄ファイル `'%1!'` をオープンできません。

原因　bcp が破棄ファイルを開くことができません。

対処法　指定されたファイルを確認します。

メッセージ 53 : 破棄ファイルの書き込みエラー

メッセージ・タイプ エラー

記号定数 BERRWDF

メッセージ bcp 破棄ファイル '%1!' の書き込み中に I/O エラーが発生しました。

原因 bcp 破棄ファイルへの書き込みでエラーが発生しました。

対処法 指定されたファイルを確認します。

メッセージ 54 : ファイルをクローズできない

メッセージ・タイプ エラー

記号定数 BERRECF

メッセージ ファイル '%1!' を閉じることができません。データがコピーされなかった可能性があります。

原因 ファイルを閉じるときにエラーが発生しました。

対処法 指定されたファイルを確認します。

メッセージ 55 : バッチ・サイズの調整

メッセージ・タイプ 警告

記号定数 BERRDISCBATWAR

メッセージ 警告： 破棄ファイル機能の最適化のために、バッチ・サイズが値 '%1!' に調整されました。

原因 最大メモリ使用量を超えたため、配列サイズが縮小されました。

メッセージ 56 : 最大ロー数への到達

メッセージ・タイプ エラー

記号定数 BERRMAXROWNUM

メッセージ bcp が処理可能な最大ロー数に達しました。合計で '%1!' ローが処理されて、 bcp オペレーションは終了しました。

原因 bcp が最大ロー数を処理して終了しました。

defncopy メッセージ

メッセージ 1 : メモリ割り当ての失敗

メッセージ・タイプ　エラー

記号定数　ERRNOMEM

メッセージ　致命的なエラー：メモリの割り当てに失敗しました。

原因　Client-Library アプリケーションがメモリを割り当てることができません。

対処法　アプリケーションで使用可能なメモリを確認します。物理メモリまたは仮想メモリを増やすか、他のアプリケーションを終了してメモリを解放します。

メッセージ 2 : 読み込み領域の不足

メッセージ・タイプ　エラー

記号定数　ERRNOREADSPACE

メッセージ　I/O エラー：領域不足のためデータを入力できません。

原因　読み込みバッファ用の領域が不足しています。

対処法　入力バッファを確認します。

メッセージ 3 : 入力ファイルを開けない

メッセージ・タイプ　エラー

記号定数　ERRNOINFILE

メッセージ　入力ファイル '%1!' をオープンできません。

原因　defncopy が入力用のデータ・ファイルを開くことができません。

対処法　指定されたファイルを確認します。

メッセージ 4 : 出力ファイルを開けない

メッセージ・タイプ エラー

記号定数 ERRNOOUTFILE

メッセージ 出力ファイル '%1!' をオープンできません。

原因 defncopy が出力用のデータ・ファイルを開くことができません。

対処法 指定されたファイルを確認します。

メッセージ 5 : 不正な引数

メッセージ・タイプ エラー

記号定数 ERRBADARG

メッセージ defncopy : 不明なパラメータ '%1!'。

原因 不明なパラメータが発行されました。

対処法 有効なパラメータを指定します。

メッセージ 6 : ファイルがフラッシュされない

メッセージ・タイプ エラー

記号定数 ERRNOFLUSH

メッセージ ファイル '%1!': '%2!' をフラッシュできませんでした。

原因 オペレーティング・システムが指定されたファイルのフラッシュに失敗しました。

メッセージ 7 : 予期しないオブジェクトの定義

メッセージ・タイプ エラー

記号定数 ERRNOOBJDEF

メッセージ オブジェクト '%1!' の定義が見つかりません。

メッセージ 8 : 異常終了

メッセージ・タイプ　エラー

記号定数　ERRABORT

メッセージ　`defncopy` がアボートされました。

原因　割り込みハンドラがトリガされました。

メッセージ 9 : 無効な方向

メッセージ・タイプ　エラー

記号定数　ERRBADDIRECTION

メッセージ　(方向は「イン」か「アウト」のどちらかです)

原因　指定された方向が無効です。

対処法　方向を修正します。

メッセージ 10 : オブジェクト名がない

メッセージ・タイプ　エラー

記号定数　ERRNOOBJNAME

メッセージ　(少なくとも `1` つのオブジェクト名が必要です)

原因　オブジェクト名が指定されていません。

対処法　少なくとも 1 つのオブジェクト名を指定します。

isql メッセージ

メッセージ 1 : メモリ割り当ての失敗

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_MALLOC

メッセージ 致命的なエラー：メモリの割り当てに失敗しました。

原因 Client-Library アプリケーションがメモリを割り当てることができません。

対処法 アプリケーションで使用可能なメモリを確認します。物理メモリまたは仮想メモリを増やすか、他のアプリケーションを終了してメモリを解放します。

メッセージ 8 : データベース名の長さ

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_LONGDBNAME

メッセージ データベース名が長すぎます。

原因 指定されたデータベース名が長すぎます。

対処法 有効な長さのデータベース名を指定します。

メッセージ 9 : CS-Lib メッセージ・コールバック・ルーチンのインストール

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_CSMSGCB

メッセージ `CS-Library` メッセージのコールバック・ルーチンをインストールできません。

原因 isql がエラー・ハンドラをインストールできません。

メッセージ 10 ： CT-Lib の初期化

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_INITCTLIB

メッセージ　`Client-Library` を初期化できません。

原因　ct_init 関数の呼び出しが失敗しました。

対処法　アプリケーションを再起動します。

メッセージ 11 ： CT-Lib メッセージ・コールバック・ルーチンのインストール

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_CTMSGCB

メッセージ　`CS-Library` クライアント・メッセージのコールバック・ルーチンをインストールできません。

原因　ct_callback 関数の呼び出しが失敗しました。

対処法　アプリケーションを再起動します。

メッセージ 12 ：サポートされないデータ型

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_DATATYPE

メッセージ　サポート対象外のデータ型を検出しました。

原因　無効なデータ型が指定されました。

対処法　無効な引数を修正します。

メッセージ 13 ：バッファのオーバフロー

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_BUFFOVFLW

メッセージ　ローの印刷中にバッファ・オーバフローが発生しました。

原因　印刷データが多すぎます。

メッセージ 15 : 無効なメモリ・ブロック・サイズ

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_INVMEMBLCK

メッセージ メモリ・ブロック・サイズの指定が無効です。

原因 Client-Library アプリケーションがメモリを割り当てることができません。

対処法 アプリケーションで使用可能なメモリを確認します。

メッセージ 16 : 無効なメモリ・ハンドル

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_INVMEMHNDL

メッセージ メモリ・ハンドルの指定が無効です。

原因 Client-Library アプリケーションがメモリを割り当てることができません。

対処法 アプリケーションで使用可能なメモリを確認します。

メッセージ 17 : 内部メモリの割り当てエラー

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_INTMALLOC

メッセージ 内部 `isql` メモリ割り当てエラー

原因 Client-Library アプリケーションがメモリを割り当てることができません。

対処法 アプリケーションで使用可能なメモリを確認します。

メッセージ 18 : エディタ・コマンドが長すぎる

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_LONGEDCMDLN

メッセージ　エディタを呼び出すコマンド・ラインが長すぎます。

原因　エディタを起動するために呼び出されたコマンドが長すぎます。

対処法　コマンドを有効な長さに短縮します。

メッセージ 19 : 初期化されていないアプリケーション・コンテキスト

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_APPCONTEXT

メッセージ　`isql` アプリケーション・コンテキストが初期化されていません。

原因　ct_config 関数の呼び出しが失敗しました。

対処法　アプリケーションを再起動します。

メッセージ 20 : 接続の失敗

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_CONFAILED

メッセージ　サーバとの接続が確立されていません。

原因　ct_connect 関数の呼び出しが失敗しました。

対処法　サーバへの接続を再度試みます。

メッセージ 21 : コマンド・ハンドルがない

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_NOCMDHNDL

メッセージ　利用できるコマンド・ハンドルがありません。

原因　コマンド・ハンドルがありません。

対処法　アプリケーションを再起動します。

メッセージ 23 : ファイルの位置のリセット失敗

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_RSTPOSIND

メッセージ ファイルの位置指示子を、ファイルの先頭にリセットできませんでした。

原因 オペレーティング・システムがインジケータを再配置できませんでした。

対処法 アプリケーションを再起動します。

メッセージ 24 : コマンド・バッファがクリアされない

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_CLRCMDBUF

メッセージ コマンド・バッファをクリアできませんでした。

原因 ct_cancel 関数の呼び出しが失敗しました。

対処法 アプリケーションを再起動します。

メッセージ 25 : コマンドが起動されない

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_INITCMD

メッセージ コマンドを起動できませんでした。

原因 ct_command 関数の呼び出しが失敗しました。

対処法 アプリケーションを再起動します。

メッセージ 26 : コマンド・ハンドルがクリアされない

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_CLRCMDHNDL

メッセージ コマンド・ハンドル内のコマンドをクリアできませんでした。

原因 ct_cancel 関数の呼び出しが失敗しました。

対処法 アプリケーションを再起動します。

メッセージ 28 : コマンド引数が長すぎる

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_LONGCMDLN

メッセージ　コマンド・ラインが長すぎます。

原因　コマンド引数が長すぎます。

対処法　有効な長さの引数を指定します。

メッセージ 29 : ファイル名の不足

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_FILENAME

メッセージ　ファイル名または実行プログラム名がありません。

原因　ファイル名または実行プログラム名が不足しています。

対処法　ファイルまたは実行プログラムの名前を指定します。

メッセージ 30 : プロンプト・ラベルが長すぎる

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_LONGPRMPTLBL

メッセージ　プロンプトのラベルが長すぎます。

原因　プロンプト・ラベルが長すぎます。

対処法　有効な長さのプロンプト・ラベルを指定します。

メッセージ 31 : プロンプト入力の不一致

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_DIFFINPUT

メッセージ　最初のプロンプトで入力した値と、確認用プロンプトの入力が異なっています。

原因　最初のプロンプトと確認用プロンプトへの入力が一致しません。

対処法　両方のプロンプトに同じ入力を行います。

メッセージ 32 : 引用符の不足

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_QUOTES

メッセージ 引用符付きのファイル名に、閉じる引用符がありません。

原因 ファイル名に開始の引用符はありますが、終了の引用符がありません。

対処法 不足している引用符を追加します。

メッセージ 33 : ディレクトリの作成失敗

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_PRIVDIR

メッセージ ディレクトリ '%1!' を作成できませんでした : '%2!'

原因 isql が指定されたディレクトリを作成できません。

対処法 isql コマンド履歴が含まれているディレクトリを確認します。

メッセージ 34 : 予期しない引数のタイプ

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_PRIVDIRTYPE

メッセージ '%1!' の予期しない種類が見つかりました : '%2!'

原因 isql はコマンド履歴ファイルのディレクトリ・タイプを特定できません。

対処法 isql コマンド履歴が含まれているディレクトリを確認します。

メッセージ 35 : 履歴ファイルを開けない

メッセージ・タイプ エラー

記号定数 LOC_ENBR_ERR_LOGWRITE

メッセージ '%1!' を書き込み用にオープンできませんでした : '%2!'

原因 isql はコマンド履歴ファイルを開いて書き込むことができません。

対処法 isql コマンド履歴が含まれているディレクトリを確認します。

メッセージ 36 ：テンポラリ・ファイルの削除失敗

メッセージ・タイプ　エラー

記号定数　LOC_ENBR_ERR_TMPFILEDEL

メッセージ　テンポラリ・ファイル `'%1!'` を削除できませんでした。

原因　isql が指定されたテンポラリ・ファイルを削除できません。

索引

E

F

G

H

I

K

L

M

O

P

R

S

T

U

か

こ

さ

す

せ

た

は

ひ

ふ

へ

め

も

ゆ

ら