SYBASE®

設定ガイド

Open Client™ and Open Server™

12.5.1

UNIX 版

ドキュメント ID：DC35838-01-1251-01
改訂：2003 年 11 月

マニュアルの注文

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

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

目次

はじめに

この『Open Client/Server 設定ガイド UNIX 版』では、次のプラットフォームで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ます。

- HP Tru64 UNIX
- HP 9000 HP-UX
- IBM RISC System/6000 AIX
- Linux
- Silicon Graphics IRIX
- Sun Solaris 2.x (SPARC)

各プラットフォームでサポートしているバージョンとオペレーティング・システムについては、使用しているプラットフォームの『リリース・ノート』を参照してください。

対象読者

このマニュアルは、Sybase のシステム管理者、Sybase データベース管理者、開発者を対象としています。ここでは、アプリケーションのプログラミングよりも、システム管理の点から、設定作業とトピックを説明します。

このマニュアルの内容

『Open Client/Server 設定ガイド UNIX 版』は次の 3 部によって構成されています。

- 設定手順
- 設定ユーティリティ
- 設定に関する参照情報

設定手順

- 「第 1 章 設定の概要」では、設定プロセスの概要と設定に必要な条件について説明します。
- 「第 2 章 Open Client の基本設定」では、クライアント・アプリケーションをサーバに接続する方法について説明し、必要な設定作業を列挙します。
- 「第 3 章 Open Server の基本設定」では、Open Server アプリケーションがクライアント接続要求を受信するための方法について説明し、接続に必要な設定作業を列挙します。
- 「第 4 章 Sybase フェールオーバのための Open Client の設定」では、フェールオーバ中に Open Client アプリケーションがセカンダリ・サーバに接続できるようにするための設定に必要な手順について説明します。

- 「第 5 章 ディレクトリ・サービスの使い方」では、アプリケーションがディレクトリ・サービスから設定情報を取得する方法について説明し、アプリケーションがディレクトリ・サービスを使用するのに必要な設定作業を列挙します。
- 「第 6 章 セキュリティ・サービスの使い方」では、アプリケーションがネットワークをベースとしたセキュリティ・サービスを使用する方法について説明し、必要な設定作業を列挙します。

設定ユーティリティ

- 「第 7 章 dscp の使用」では、ディレクトリ・サービスと interfaces ファイルのサーバ・エントリを設定する、dscp コマンド・ライン・ユーティリティを使用する方法について説明します。
- 「第 8 章 dsedit の使用」では、dsedit ユーティリティを使用して、ディレクトリ・サービスと *interfaces* ファイルのサーバ・エントリを設定する方法について説明します。dsedit はグラフィカル・ユーザ・インタフェースを使用した Windows ユーティリティです。

設定に関する参照情報

設定トピックは、設定情報のソースごとに付録として分類されています。

- 「付録 A 環境変数」では、Open Client/Server 製品で使用する環境変数をリストし、その設定方法について説明します。
- 「付録 B 設定ファイル」では、設定ファイルの概要を示し、次の点について説明します。
  - *libtcl.cfg* (ドライバ設定ファイル)
  - *interfaces* (interfaces ファイル)
  - *ocs.cfg* (ランタイム設定ファイル)
- 「付録 C ローカライゼーション」では、ローカライゼーション・ファイルの概要を示し、次の点について説明します。
  - *locales.dat* ファイル
  - *objectid.dat* ファイル
  - *mnemonic.dat* ファイル
  - 変換設定ファイル
  - ローカライズされたメッセージ・ファイル
  - 照合順ファイル
- 「付録 D DCE セキュリティ・サービス」では、DCE セキュリティ・ドライバがサポートするセキュリティ・サービスをリストし、Open Client/Server セキュリティ・メカニズムとして使用するための DCE の設定条件について説明します。

- 「付録 E CyberSafe Kerberos セキュリティ・サービス」では、CyberSafe Kerberos セキュリティ・ドライバがサポートするセキュリティ・サービスをリストし、Open Client/Server セキュリティ・メカニズムとして使用するための CyberSafe の設定条件について説明します。
- 「付録 F Open Client/Open Server の SSL (Secure Socket Layer)」では、Open Client および Open Server の SSL サポートと、SSL (Security Socket Layer) プロトコルの使用に必要なシステム設定作業について要約します。

その他の情報ソース

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

- Getting Started CD には、PDF 形式のリリース・ノートとインストール・ガイドが収録されています。また、その他のマニュアルや、Technical Library CD には含まれない更新情報が収録されることもあります。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)。
- Technical Library CD には製品マニュアルが入っており、この CD は製品のソフトウェアに同梱されています。DynaText リーダー (Technical Library CD に収録) を使用すると、この製品に関する技術情報に簡単にアクセスできます。

  Technical Library のインストールと起動の方法については、マニュアル・パッケージに含まれている『Technical Library Installation Guide』を参照してください。
- Technical Library Product Manuals Web サイトは、Technical Library CD の HTML バージョンで、標準の Web ブラウザを使ってアクセスできます。また、製品マニュアルのほか、EBFs/Updates、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。

  Technical Library Product Manuals Web サイトにアクセスするには、Product Manuals (http://www.sybase.com/support/manuals/) にアクセスしてください。

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品認定の最新情報にアクセスする**

1. Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)
2. 左側のナビゲーション・バーから [Products] を選択します。
3. 製品リストから製品名を選択し、[Go] をクリックします。
4. [Certification Report] フィルタを選択し、時間枠を指定して [Go] をクリックします。

5 [Certification Report] のタイトルをクリックして、レポートを表示します。

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

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

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

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

Sybase EBF とソフトウェア・メンテナンス

❖ **EBF とソフトウェア・メンテナンスの最新情報にアクセスする**

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

2 [EBFs/Maintenance] を選択します。すでに Web アカウントをお持ちの場合はユーザ名とパスワードを要求されますので、各情報を入力します。Web アカウントをお持ちでない場合は、新しいアカウントを作成します。サービスは無料です。

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。

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。

表記の規則

このマニュアルでは、次の表記規則を使用します。

- 入力するコマンドは、次のように表記します。

  ```
  this font
  ```

- インストール環境に適した値に置き換える言葉や文字は、次のように表記します。

  ```
  this font
  ```

- ファイル名とディレクトリは次のように表記します。

  /usr/u/sybase

- プログラム名、ユーティリティ名、プロシージャ名、およびコマンド名は次のように表記します。

  dscp

このマニュアルでは、次の構文表記を使用します。

表 1: 構文の表記規則

構文要素	意味
command	コマンド名、コマンド・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などのキーワードは、**太字**で表記される。
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。

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

設定の概要

UNIX 用の『インストール・ガイド』の指示に従って、Open Client か Open Server をインストールしてから、このマニュアルを読んでください。Open Client は、SDK (Software Developer's Kit) の一部としてパッケージされています。

この章では、Open Client と Open Server の設定プロセスの概要を説明します。

Open Client と Open Server について

Open Client は、*dblib* と *ctlib* という 2 つの API (アプリケーション・プログラミング・インタフェース) と、Net-Library™ を提供します。これらの製品を使用することで、Adaptive Server アプリケーション、Open Server アプリケーション、サード・パーティの製品、カスタマ・アプリケーション、その他の Sybase 製品の間で通信することが可能になります。

Open Server は、カスタム・サーバの作成に必要なツールとインタフェースを提供します。Open Client と同様に、プログラミング API と Net-Library (DB-library™ を除く) がクライアントや他のサーバとの通信を可能にします。さらに Open Server は、次の機能を持つルーチンも提供します。

- 複数のクライアント接続を処理する。
- クライアントとの対話をスケジュールする。
- エラー条件を処理する。
- サーバから要求されたその他の機能を実行する。

Open Client と Open Server の詳細については、次のマニュアルを参照してください。

- 『Open Client Client-Library リファレンス・マニュアル』
- 『Open Client DB-Library リファレンス・マニュアル』
- 『Open Server Server-Library リファレンス・マニュアル』

設定の概要

Open Client/Open Server ソフトウェアを正しく機能させるには、特定の情報が必要です。「設定」とは、この情報を使用できるようにシステムを準備するプロセスです。

Open Client と Open Server は、設定された情報を使用して次の処理を行います。

- Open Client (DB-library を除く) または Open Server アプリケーションを初期化する。
- Adaptive Server® または Open Server アプリケーションとの接続を確立する。

注意 注意書きがある場合を除いて、このマニュアルの内容は DB-Library と Client-Library™ の両方に適用されます。

特に DB-Library は初期ローカライゼーション値を決定するのに環境変数を使用せず、*libtcl.cfg* ファイルを調べません。ただし、SYBASE 環境変数と DSQUERY 環境変数は調べます。

DB-Library 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。

初期化プロセス

アプリケーションを初期化するために、Open Client と Open Server は次のアクションを行います。

- SYBASE 環境変数を使用して Sybase インストール・ディレクトリのロケーションを決定する。
- ロケール固有 POSIX 環境変数 LC_*、LANG、LC_ALL、LC_COLLATE、および *locales.dat* ファイルを使用して、アプリケーションがどの言語、文字セット、照合順を使用するかを決定する。
- *libtcl.cfg* ファイルを使用して、必要に応じてディレクトリ・ドライバ、セキュリティ・ドライバ、ネットワーク (Net-Library) ドライバをロードする。

接続プロセス

クライアントとサーバは「接続」を介してお互いに通信します。クライアント・アプリケーションがサーバ・アプリケーションに接続するには、サーバ・アプリケーションがクライアントの接続要求を受信していなければなりません。

接続するために、Open Client は次のアクションを行います。

- DSQUERY 環境変数を使用して、ターゲット・サーバの名前を決定する。DSQUERY は Open Client アプリケーションがターゲット・サーバの名前を指定していない場合にだけ使用します。DSQUERY とアプリケーションの両方で指定した場合、アプリケーションの指定が優先されます。
- ディレクトリ・サービスと *interfaces* ファイルを使用してターゲット・サーバのアドレスを取得する。

注意　DB-library は *interfaces* ファイルを使用してサーバのみ検索できます。

接続要求を受信するために、Open Server は次のアクションを行います。

- DSLISTEN 環境変数を使用して Open Server アプリケーションの名前を決定する。
- *interfaces* ファイルとディレクトリ・サービスを使用して、Open Server アプリケーションのアドレスを指定する。

注意　DSLISTEN は Open Server アプリケーションが初期化時にサーバを指定していない場合にだけ使用します。

設定作業

Open Client/OpenServer 製品がアプリケーションを初期化して接続を行う前に、いくつかの基本的な設定作業を行います。

- ターゲットのデフォルト・サーバと初期ローカライゼーション値を指定するように、環境変数を設定する。Open Client と Open Server アプリケーションがサーバの名前を明示的に指定していない場合は、DSQUERY と DSLISTEN の値が使用されます。
- ターゲット・サーバのアドレスが使用可能かどうかを確認する。
- 必要であれば、ネットワーク・ドライバを設定する。

次のいずれかを使用する場合は、追加作業が必要です。

- ディレクトリ・サービス
- セキュリティ・サービス
- 初期ローカライゼーション値とカスタム・ローカライゼーション値、または初期ローカライゼーション値の代わりにカスタム・ローカライゼーション値

第 2 章以降では、設定手順を説明します。それぞれのインストール環境に該当する章を参照してください。

第 2 章

Open Client の基本設定

この章では、Open Client に必要な基本の設定を説明します。

注意 注意書きがある場合を除いて、この章の内容は DB-Library と Client-Library の両方に適用されます。

特に DB-Library は初期ローカライゼーション値を決定するのに環境変数を使用せず、*libtcl.cfg* ファイルを調べません。ただし、SYBASE 環境変数と DSQUERY 環境変数は調べます。

DB-Library 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。

Open Client の設定の概要

すべての Open Client アプリケーションは、次のような基本設定情報を必要とします。これらの情報は、初期化時と接続時に取得されます。

注意 項目 1 ～ 3 (DB-Library には適用されません) は、Open Client Client-Library アプリケーションが cs_ctx_alloc または cs_ctx_global ルーチンを呼び出す場合に行われます。項目 4 ～ 6 は、Open Client アプリケーションが ct_connect を呼び出す場合に行われます。

1 SYBASE 環境変数に定義されている Sybase インストール・ディレクトリのロケーション。

2 ロケール名。Open Client は次の POSIX 環境変数の値をロケール名として使用します。

- LC_ALL
- LANG (LC_ALL が定義されていない場合)

Open Client はあとからこの値を使用して *locales.dat* ファイルからローカライゼーション情報を取得します。どちらの環境変数も定義されていない場合は、Open Client はロケール名として「デフォルト」を使用します。

3 ローカライズされたメッセージ・ファイルと文字セット・ファイル。Open Client は、*locales.dat* ファイルを調べて、上記の手順で指定したロケール名と一致するエントリを探し、*locales.dat* ファイルに設定されているローカライズされたメッセージ・ファイルと文字セット・ファイルをロードします。

4 ターゲット・サーバの名前。Open Client は、いずれかのソースからこの順序でターゲット・サーバの名前を取得します。

   a ct_connect (または dbopen) に対する呼び出しにサーバ名を指定できるクライアント・アプリケーション。isql などのアプリケーションの中には、コマンド・ライン・オプションを使用してターゲット・サーバの名前を指定できるものもあります。

   b アプリケーションがターゲット・サーバを指定していない場合、DSQUERY 環境変数。

   c DSQUERY が設定されていない場合は、デフォルト名の SYBASE。

5 ターゲット・サーバのネットワーク・アドレス。Open Client は、ディレクトリ・サービスまたは *interfaces* からターゲット・サーバのアドレスを取得します。DB-Library は *libtcl.cfg* ファイルを調べず、*interfaces* ファイルにアクセスします。

   - ディレクトリ・サービス − Open Client は *libtcl.cfg* ファイルの [DIRECTORY] セクション内のエントリを探して、サーバ・アドレス情報をどこで調べるかを決定します。CS_DS_PROVIDER プロパティの設定値によって、アプリケーションがどの [DIRECTORY] エントリを検索するかが決定されます。プロパティが設定されていない場合は、[DIRECTORY] セクションの最初のエントリがデフォルトで使用されます。

   - *interfaces* − ディレクトリ・サービスが使用されていない、または使用されていても機能していない場合は、Open Client は *interfaces* ファイルを調べて、前の手順で指定した名前と一致する SERVERNAME を検出し、それに対応するターゲット・アドレスを使用します。

6 ネットワーク・ドライバの名前 (DB-Library には適用されません)。Open Client は、*libtcl.cfg* の [DRIVERS] セクションを調べて、どのネットワーク・ドライバをロードするかを決定します。

   表 7-2 (43 ページ) の「トランスポート・タイプ」は、有効なネットワーク・プロトコルを示します。この値は、どのネットワーク・ドライバをロードするかを決定します。「ネットワーク・ドライバの追加」(72 ページ) では、ネットワーク・ドライバを追加する手順を説明します。

7　セキュリティ・サービス・ドライバの名前 (DB-Library には適用されません)。Open Client は、*libtcl.cfg* の [SECURITY] セクションを調べて、どのセキュリティ・ドライバをロードするかを決定します。

セキュリティ・サービスの詳細については、「第 6 章 セキュリティ・サービスの使い方」を参照してください。

Adaptive Server バージョン 12.5 では、以前のバージョンで格納されたデータとは異なる制限を持つデータを格納できます。CS_VERSION_125 が init 時に設定される場合、CT-Library クライアントは新しい制限を使用できますが、DB-Library クライアントは使用できません。また、クライアントも、データが使用できる新しい制限を処理できる必要があります。Open Client バージョン 12.5.1 は、Adaptive Server 12.5.1 の制限をサポートします。Open Client と Open Server の旧バージョンを使用している場合は、次の処理を行ったときにデータを処理できません。

- Adaptive Server バージョン 12.5.1 へのアップグレード
- 幅広いカラムを持つテーブルの削除と再作成
- 長いデータの挿入

Open Client の設定作業

Open Client が正しくクライアント・アプリケーションを初期化して接続要求を実行するには、次の作業を行ってください。

1　次のように、環境変数を設定します。

LC_ALL または LANG 環境変数を任意のロケール名に設定します。指定するロケール名は、*locales.dat* ファイルのエントリに対応させてください。

LC_ALL または LANG を設定しない場合は、*locales.dat* の「デフォルト」エントリがアプリケーションで使用するローカライゼーション値を反映していることを確認してください。

環境変数の設定方法については、「付録 A 環境変数」を参照してください。

2　ローカライゼーション・ファイルを次のように設定します。

locales ファイルに指定されている言語、文字セット、照合順と一致するローカライゼーション・ファイルがあることを確認してください。

アプリケーションが「カスタム・ローカライゼーション値」を使用する場合は、LC_ALL、LC_COLLATE、LC_TYPE、LC_MESSAGE、または LC_TIME 環境変数をロケール名に設定します。アプリケーションがどの環境変数を使用するかわからない場合は、すべての環境変数を希望のロケール名に設定してください。

3 DSQUERY 環境変数をターゲット・サーバの名前に設定します。

クライアント・アプリケーションにターゲット・サーバの名前が指定されている場合、DSQUERY を設定する必要はありません。DSQUERY が設定されていなくて、アプリケーションにもサーバ名が指定されていない場合には、Open Client はサーバ名として “SYBASE” を使用します。

ローカライゼーションについては、「付録 C ローカライゼーション」を参照してください。

4 デフォルト値を変更する場合、*libtcl.cfg* を次のように設定します。

- *libtcl.cfg* の [DRIVERS] セクションにネットワーク・トランスポート・ドライバを指定します。
- *libtcl.cfg* の [DIRECTORY] セクションにディレクトリ・ドライバを指定します。
- *libtcl.cfg* の [SECURITY] セクションにセキュリティ・ドライバを指定します。

libtcl.cfg については、「付録 B 設定ファイル」を参照してください。

5 *interfaces* ファイルまたはディレクトリ・サービスを次のように設定します。

dscp を使用して、*interfaces* または DCE ディレクトリ・サービスにサーバ・エントリを作成します。

dscp の使い方については、「第 7 章 dscp の使用」を参照してください。

interfaces については、「interfaces ファイル」(73 ページ) を参照してください。

ディレクトリ・サービスについては、「第 5 章 ディレクトリ・サービスの使い方」を参照してください。

6 dscp を使用して、*interfaces* または DCE ディレクトリ・サービスにサーバ・エントリを作成します。

dscp の使い方については、「第 7 章 dscp の使用」を参照してください。

interfaces については、「interfaces ファイル」(73 ページ) を参照してください。

ディレクトリ・サービスについては、「第 5 章 ディレクトリ・サービスの使い方」を参照してください。

Client-Library の互換性

UNIX プラットフォームの Client-Library 12.5.1 は、表 2-1 に示されている Open Server と Adaptive Server 製品で稼働することが保証されています。

注意　特定のプラットフォームまたは OS レベルの情報については、http://www.sybase.com にある各製品の Certification Report を参照してください。

表 2-1: Open Client の互換性

Open Client 12.5.1 (SDK 12.5.1) プラットフォーム	Open Server 12.5.1	Open Server 12.5	Open Server 12.0	Adaptive Server 12.5.1	Adaptive Server 12.5	Adaptive Server 12.0
HP Tru64 UNIX	x	x	x	x	x	x
HP 9000/800 HP-UX 11.0	x	x	x	x	x	x
HP 9000/800 Itanium	x	なし	なし	x	なし	なし
IBM RS/6000 AIX 4.3.3 **注意** • AIX 4.3.3 の 64 ビット・ライブラリ SDK/OS は、AIX 5.1 と互換性がなく動作しない。 • AIX 4.3.3 の 32 ビット・ライブラリは、バージョン 12.5、ESD #6 以降について AIX 5.1 と互換性があり AIX 5.1 上で動作する。	x	x	x	x	x	x
IBM AIX 5.1	x	x	x	x	x	なし
Linux (32 ビット版)	x	x	なし	x	x	なし
Linux (64 ビット版)	x	x	なし	x	x	なし
Linux Itanium (64 ビット版)	x	なし	なし	x	なし	なし
Mac OS X 10.1	なし	x	なし	なし	なし	なし
Mac OS X 10.2	なし	x	なし	x	なし	なし
SGI IRIX 6.5 (32 ビット版)	x	x	x	x	x	x
SGI IRIX 6.5 (64 ビット版)	x	x	なし	x	なし	なし
Solaris 2.8 (SPARC)	x	x	x	x	x	x

記号の説明：**x** = 互換性あり、**なし** = このプラットフォーム版の製品がない

このほかに、Open Client/C の互換性に関する次の問題に注意してください。

- アプリケーションを構築するために使用するライブラリは、そのアプリケーションのコンパイルに使用するライブラリと同じバージョン・レベルでなければならない。
- アプリケーションに含まれるヘッダ・ファイルは、アプリケーションがリンクしているライブラリと同じバージョン・レベルでなければならない。

第 3 章

Open Server の基本設定

この章では、Open Server に必要な基本の設定について説明します。

Open Server アプリケーションについて

Open Server アプリケーションは、機能的に次の 3 つのタイプに分けられます。

- スタンドアロン
- 補助
- ゲートウェイ

Open Server アプリケーションの設定は、アプリケーションのタイプによって異なります。Open Server アプリケーションのタイプの詳細については、『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。

Open Server の設定の概要

すべての Open Server アプリケーションは、次のような基本設定情報を必要とします。これらの情報は、初期化時と接続時に取得されます。

1 SYBASE 環境変数に定義されている Sybase インストール・ディレクトリのロケーション。

2 ロケール名。Open Server は次の POSIX 環境変数の値をロケール名として使用します。

- LC_ALL
- LANG (LC_ALL が定義されていない場合)

Open Server はあとでこの値を使用して *locales.dat* ファイルからローカライゼーション情報を取得します。どちらの環境変数も定義されていない場合は、Open Server はロケール名として「デフォルト」を使用します。

3 ローカライズされたメッセージ・ファイルと文字セット・ファイル。Open Server は、*locales.dat* ファイルを調べて、名前が上記の手順 2 で指定したロケール名と一致するエントリを探し、*locales.dat* ファイルに指定されているローカライズされたメッセージ・ファイルと文字セット・ファイルをロードします。

4 ターゲット・サーバの名前。Open Server は、次のソースのいずれかからこの順序で Open Server アプリケーションの名前を取得します。

- srv_init に対する呼び出しにサーバ名を指定できる Open Server アプリケーション
- アプリケーションにターゲット・サーバ名が指定されていない場合、DSLISTEN 環境変数
- DSLISTEN が設定されていない場合、デフォルト名の SYBASE

5 ターゲット・サーバのネットワーク・アドレス。Open Server は、ディレクトリ・サービスまたは *interfaces* からターゲット・サーバのアドレスを取得します。

ディレクトリ・サービス － Open Server は *libtcl.cfg* ファイルの [DIRECTORY] セクション内のエントリを探して、サーバ・アドレス情報をどこで調べるかを決定します。CS_DS_PROVIDER プロパティの設定値によって、アプリケーションがどの [DIRECTORY] エントリを検索するかが決定されます。プロパティが設定されていない場合は、[DIRECTORY] セクションの最初のエントリがデフォルトで使用されます。

interfaces － ディレクトリ・サービスが使用されていない、または使用されていても機能していない場合は、Open Server は *interfaces* ファイルを調べて、上記の手順で指定した名前と一致する SERVERNAME を検出し、それに対応するターゲット・アドレスを使用します。

6　ネットワークベースのセキュリティ・サービスを使用する接続をクライアントが要求している場合は、Open Server は *libtcl.cfg* の [SECURITY] セクションで該当するセキュリティ・ドライバを探します。

Open Server の互換性

Open Server 12.5.1 は、表 3-1 に示されている Client-Library/C と Adaptive Server 製品で稼働することが保証されています。

表 3-1: Open Server の互換性

Open Server 12.5.1 のプラットフォーム	Client-Library 12.5.1	Client-Library 12.5	Client-Library 12.0	Adaptive Server 12.5.1	Adaptive Server 12.5	Adaptive Server 12.0
HP Tru64 UNIX	x	x	x	x	x	x
HP 9000/800 HP-UX 11.0	x	x	x	x	x	x
HP 9000/800 Itanium	x	なし	なし	x	なし	なし
IBM AIX 5.1	x	x	x	x	なし	なし
IBM RS/6000 AIX 4.3.3 **注意** • AIX 4.3.3 の 64 ビット・ライブラリ SDK/OS は、AIX 5.1 と互換性がなく動作しない。 • AIX 4.3.3 の 32 ビット・ライブラリは、バージョン 12.5、ESD #6 以降について AIX 5.1 と互換性があり AIX 5.1 上で動作する。	x	x	x	x	x	x
Linux	x	x	なし	x	x	なし
Linux Itanium	x	なし	なし	x	なし	なし
Mac OS X	なし	x	なし	x	なし	なし
SGI IRIX 6.5 (32 ビット)	x	x	x	x	x	x
SGI IRIX 6.5 (64 ビット版)	x	なし	なし	x	なし	なし
Solaris 2.8 (SPARC)	x	x	x	x	x	x

記号の説明：x = 互換性あり、**なし** = このプラットフォーム版の製品がない

さらに、Open Server については、次の互換性の問題にも注意してください。

- アプリケーションに含まれるヘッダ・ファイルは、アプリケーションがリンクしているライブラリと同じバージョン・レベルでなければならない。
- アプリケーションを構築するために使用するライブラリは、そのアプリケーションのコンパイルに使用するライブラリと同じバージョン・レベルでなければならない。

- Bulk-Library のルーチンは、Open Server バージョン 12.x のルーチンを呼び出すアプリケーションでは使用できない。
- バージョン 11.x 以降、Open Server コード内の DB-Library/C アプリケーションはサポートされていない。

設定作業

Open Server が正しくサーバ・アプリケーションを初期化して接続要求に応答するには、次の作業を行ってください。

1 *libtcl.cfg* を次のように設定します。

- *libtcl.cfg* の [DIRECTORY] セクションにディレクトリ・トランスポート・ドライバを指定します。
- *libtcl.cfg* の [SECURITY] セクションにセキュリティ・ドライバを指定します。

libtcl.cfg については、「付録 B 設定ファイル」を参照してください。

2 *interfaces* ファイルまたはディレクトリ・サービスを次のように設定します。

dscp を使用して、*interfaces* または DCE ディレクトリ・サービスにサーバ・エントリを作成します。

dscp の使い方については、「第 7 章 dscp の使用」を参照してください。*interfaces* の詳細については、「interfaces ファイル」(73 ページ) を参照してください。ディレクトリ・サービスについては、「第 5 章 ディレクトリ・サービスの使い方」を参照してください。

3 次のように、環境変数を設定します。

- LC_ALL または LANG 環境変数を任意のロケール名に設定します。

  指定するロケール名は、*locales.dat* ファイルのエントリに対応させてください。LC_ALL または LANG を設定しない場合は、*locales.dat* の「デフォルト」エントリがアプリケーションで使用するローカライゼーション値を反映していることを確認してください。

  locales に指定されている言語、文字セット、照合順と一致するローカライゼーション・ファイルがあることを確認してください。

- アプリケーションが「カスタム・ローカライゼーション値」を使用する場合は、LC_ALL、LC_COLLATE、LC_TYPE、LC_MESSAGE、または LC_TIME 環境変数をロケール名に設定します。

  アプリケーションがどの環境変数を使用するかわからない場合は、すべての環境変数を希望のロケール名に設定してください。

- DSLISTEN 環境変数を Open Server アプリケーションの名前に設定します。

  アプリケーションに Open Server アプリケーションの名前が指定されている場合、DSLISTEN を設定する必要はありません。DSLISTEN が設定されていなくて、アプリケーションにもサーバ名が指定されていない場合には、Open Server はサーバ名として “SYBASE” を使用します。

- Open Server アプリケーションがゲートウェイ・アプリケーションとして機能する場合は、DSQUERY 環境変数をターゲット・サーバの名前に設定してください。

環境変数の設定方法については、「付録 A 環境変数」を参照してください。ローカライゼーションについては、「付録 C ローカライゼーション」を参照してください。

第 4 章

Sybase フェールオーバのための Open Client の設定

Sybase の フェールオーバ機能の詳細については、『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。この章では、フェールオーバの間にセカンダリ・コンパニオンに接続するよう Open Client アプリケーションを設定する場合に必要な手順について説明します。この情報は、上記のマニュアルには含まれていません。

注意 DB-Library は HA (高可用性) フェールオーバをサポートしていません。Embedded SQL/C と Embedded SQL/COBOL はバージョン 12.5 以降、HA フェールオーバをサポートしています。

interfaces ファイルへの hafailowver 行の追加

プライマリ・コンパニオンがクラッシュしたり、shutdown または shutdown with nowait を発行して、フェールオーバが発生したりした場合は、フェールオーバ・プロパティのあるクライアントは、セカンダリ・コンパニオンに自動的に再接続します。クライアントにフェールオーバ・プロパティを付与するには、*interfaces* ファイルに “hafailover” という行を追加し、クライアントがセカンダリ・コンパニオンに接続するのに必要な情報を提供してください。この行を追加するには、ファイル・エディタか dsedit ユーティリティを使用します。

以下の *interfaces* ファイル・エントリは、プライマリ・コンパニオン “PERSONNEL1” とセカンダリ・コンパニオン “MONEY1” を非対称型に設定するためのものです。これには hafailover エントリが含まれていて、“PERSONNEL1” に接続しているクライアントがフェールオーバ時に “MONEY1” に再接続できるようになっています。

```
PERSONNEL1

    master tcp ether <hostname><port #>
    query tcp ether <hostname><port #>
hafailover <servername>
```

注意 クライアント・アプリケーションは、フェールオーバによって送信できなかったクエリを再送する必要があります。また、カーソル宣言などの接続固有のその他の情報は、リストアする必要があります。

Client-Library アプリケーションの変更

注意 クラスタにインストールされているアプリケーションは、プライマリ・コンパニオンとセカンダリ・コンパニオンの両方で実行できる必要があります。並列設定が必要なアプリケーションをインストールする場合は、フェールオーバの間にセカンダリ・コンパニオンがアプリケーションを実行できるように、セカンダリ・コンパニオンにも並列処理の設定を行う必要があります。

Client-Library 呼び出しで記述されたアプリケーションを、フェールオーバ・ソフトウェアで実行できるようにするには、変更が必要です。

❖ Client-Library 呼び出しを使用してアプリケーションを変更する

1 ct_config と ct_con_props Client-Library API 呼び出しを使用して、CS_HAFAILOVER プロパティを設定します。プロパティの有効値は CS_TRUE と CS_FALSE です。デフォルト値は CS_FALSE です。次のコードを使用して、コンテキストまたは接続レベルのどちらかにこのプロパティを設定できます。

```
CS_INT true = CS_TRUE;
CS_INT false = CS_FALSE;
retcode = ct_config(context, CS_SET, CS_HAFAILOVER, &true,
CS_UNUSED, NULL);
retcode = ct_con_props(connection, CS_SET, CS_HAFAILOVER,
&false, CS_UNUSED, NULL);
```

2　フェールオーバ・メッセージの処理コンパニオンが落ち始めると、クライアントはフェールオーバがまもなく発生するという情報メッセージを受信します。これは、クライアント・エラー・ハンドラの情報メッセージとして扱ってください。

3　フェールオーバ設定を確認します。フェールオーバ・プロパティを設定し、*interfaces* ファイルにセカンダリ・コンパニオン・サーバの有効なエントリが設定されていると、接続はフェールオーバ接続になり、クライアントは適切に再接続します。

ただし、CS_FAILOVER プロパティが設定されていても、*interfaces* ファイルに HAFAILOVER サーバのエントリがない場合 (またはその逆) は、フェールオーバ接続にはなりません。この場合は、フェールオーバ・プロパティがオフになった、高可用性ではない通常の接続になります。フェールオーバ・プロパティを確認して、接続がフェールオーバ接続かどうかを確認してください。これを行うには、CS_GET の *action* とともに ct_con_props を呼び出します。

4　リターン・コードを検証します。フェールオーバが成功したら、ct_results と ct_send を呼び出して、CS_RET_HAFAILOVER を返します。同期接続では、API 呼び出しは CS_RET_HAFAILOVER を直接返します。非同期接続では、API は CS_PENDING を返し、コールバック機能は CS_RET_HAFAILOVER を返します。リターン・コードによっては、next コマンドを送信して実行するなど、アプリケーションは必要なプロセスを行います。

5　オプション値をリストアします。クライアントがプライマリ・コンパニオンから切断されると、このクライアント接続に合わせて設定してある set オプション (たとえば、set role など) は失われます。フェールオーバした接続で、これらのオプションをリセットします。

6　アプリケーションを再構築し、フェールオーバ・ソフトウェアに含まれるライブラリにリンクさせます。

注意　sp_companion resume を発行するまでは、フェールオーバ・プロパティ (たとえば、isql -Q など) にクライアントを接続することはできません。sp_companion prepare_failback を発行してからクライアントを再接続しようとすると、sp_companion resume を発行するまでクライアントはハングします。

Sybase HA フェールオーバでの isql の使い方

isql を使用してフェールオーバ機能のあるプライマリ・サーバに接続するには、次の手順に従います。

- *interfaces* エントリで指定されているセカンダリ・コンパニオン・サーバのあるプライマリ・サーバを選択します。
- -Q コマンド・ライン・オプションを使用します。

interfaces ファイルに、「interfaces ファイルへの hafailowver 行の追加」に示されているエントリ例がある場合は、次のように入力して、フェールオーバで isql を使用できます。

```
isql -S PERSONNEL1 -Q
```

.

第 5 章

ディレクトリ・サービスの使い方

Client-Library と Server-Library アプリケーションはディレクトリ・サービスを使用して、サーバに関する情報を記録します。この章では、ディレクトリ・サービスの実行方法と、ディレクトリ・サービスに必要な設定作業について説明します。

注意 DB-Library はディレクトリ・サービスをサポートしていません。

ディレクトリ・サービスの概要

「ディレクトリ・サービス」では、ネットワーク・エンティティについての情報の作成、変更、検索を管理します。Client-Library と Server-Library アプリケーションは *interfaces* のかわりにディレクトリ・サービスを使用してサーバについての情報を取得できます。

ディレクトリ・サービスを使用する利点は、新しいサーバをネットワークに追加するときやサーバを新しいアドレスに移動するときに複数の *interfaces* ファイルを更新する必要がない点です。

UNIX プラットフォームでは、DCE (Distributed Computing Environment) で提供される CDS (Cell Directory Service) や LDAP (Lightweight Directory Access Protocol) ディレクトリ・サービスを使用できます。

LDAP ディレクトリ・サービス

LDAP は、ディレクトリ・リストへのアクセスに使用します。ディレクトリ・リストやサービスは、ネットワーク上のユーザとリソースの名前、プロファイル情報、マシン・アドレスを提供します。ユーザ・アカウントとネットワーク・パーミッションを管理するのに、これを使用できます。

LDAP サーバは一般的には階層構造で、高速なリソースの検索が可能です。従来の Sybase *interfaces* ファイルの代わりに、LDAP を使用して Sybase サーバの情報を保管したり検索したりできます。

実際のサーバや他の LDAP サービスへのゲートウェイかどうかにかかわらず、どのタイプの LDAP サーバでも LDAP サーバと呼ばれます。LDAP ドライバは LDAP クライアント・ライブラリを呼び出して、LDAP サーバへの接続を確立します。LDAP ドライバとクライアント・ライブラリは、暗号化を有効にするかどうかなどの通信プロトコル、およびクライアントとサーバの間で交換されるメッセージのコンテンツを定義します。メッセージとは、データ・フォーマット情報も含めたクライアントの読み込み、書き込み、クエリ、サーバ応答などの要求です。

LDAP ディレクトリ・サービスと Sybase *interfaces* ファイルの違い

LDAP ディレクトリ・サービスは、通常の Sybase *interfaces* ファイルの代わりとなるものです。Sybase *interfaces* ファイルでは、「フラット」ファイルにサーバ情報を格納しています。*interfaces* ファイルのサーバ情報を変更するときは、サイトの全マシン (クライアントとサーバ) を更新する必要があります。

表 5-1 は、Sybase *interfaces* ファイルと LDAP サーバの相違点を示します。

表 5-1: interfaces ファイルと LDAP ディレクトリ・サービスの比較

interfaces ファイル	ディレクトリ・サービス
プラットフォーム固有	プラットフォームに依存しない
各 Sybase インストール固有	集中階層型
別のマスタ・エントリとクエリ・エントリを含む	クライアントとサーバの両方がアクセスするサーバごとに 1 エントリを含む
サーバのメタデータを格納できない	サーバのメタデータを格納できる

従来の *interfaces* ファイルは、TCP 接続の UNIX マシンおよびフェールオーバ・マシンで次のように表示されます。

```
master tcp ether huey 5000
query tcp ether huey 5000
hafailover secondary
```

次の例は、TCP 接続の LDAP エントリとフェールオーバ・マシンを示します。

```
dn: sybaseServername=foobar, dc=sybase,dc=com
objectClass: sybaseServer
sybaseVersion: 12500
sybaseServername: foobar
sybaseService: ASE
sybaseStatus: 4
sybaseAddress: TCP#1#foobar 5000
sybaseRetryCount: 12
sybaseRetryDelay: 30
sybaseHAServernam: secondary
```

LDAP ディレクトリ・サービスへのすべてのエントリは、エンティティと呼ばれます。各エンティティは DN (識別名) を持ち、それぞれの DN に基づいて階層ツリー構造内に格納されます。このツリーは、ディレクトリ情報ツリー (DIT) と呼ばれます。接続中に DIT ベースを指定することで、クライアント接続は LDAP サーバの検索開始位置を設定します。

表 5-2 は、有効な DIT ベースの値を示したものです。

表 5-2: Sybase LDAP エントリ定義

属性名	値のタイプ	説明
sybaseVersion	整数	サーバのバージョン番号。
sybaseServername	文字列	サーバの名前。
sybaseService	文字列	サービスの種類。Sybase Adaptive Server または Sybase SQL Server®。
sybaseStatus	整数	ステータス。1 = アクティブ、2 = 停止、3 = 失敗、4 = 不明。
sybaseAddress	文字列	アドレス文字列の各エントリは # 文字で区切る。各サーバのアドレス。次の項目を含む。 • プロトコル：TCP、NAMEPIPE、SPX DECNET (大文字と小文字を区別して入力する) • sybaseStatus の値 • アドレス：そのプロトコル・タイプの有効なアドレス **注意** dscp ユーティリティは、この属性をトランスポート・タイプとトランスポート・アドレスに分割します。
sybaseSecurity (オプション)	文字列	セキュリティ OID (オブジェクト ID)。
sybaseRetryCount	整数	この属性は、CS_RETRY_COUNT にマッピングされる。CS_RETRY_COUNT は、ct_connect がサーバ名と対応するネットワーク・アドレスのシーケンスをリトライする回数を指定する。
sybaseRetryDelay	整数	この属性は、CS_LOOP_DELAY にマッピングされる。CS_LOOP_DELAY は、ct_connect がアドレスのすべてのシーケンスをリトライするまでの遅延時間を秒単位で指定する。
sybaseHAservername (オプション)	文字列	フェールオーバ保護用のセカンダリ・サーバ。

Sybase の LDAP ディレクトリ・スキーマのリストについては、*$SYBASE/$SYBASE_OCS/config* ディレクトリを参照してください。同じディレクトリに、*sybase-schema.conf* と呼ばれるファイルもあります。このファイルには、同じスキーマですが、Netscape 固有の構文のものがあります。

上記の例では、エンティティはポート番号 5000 の TCP 接続を受信する “foobar” という名前の Adaptive Server を表しています。このエンティティには、12 (回) というリトライ回数と 30 (秒) というリトライ待ち時間も指定されています。sybaseRetryCount と sybaseRetryDelay は、それぞれ CS_RETRY_COUNT と CS_LOOP_DELAY にマップされています。Client-Library はサーバから応答があるアドレスを見つけると、Client-Library とサーバ間でログイン・ダイアログが開始されます。ログインが失敗しても、Client-Library は他のアドレスをリトライすることはありません。

最も重要なエンティティはアドレス属性です。アドレス属性には、サーバへの接続を設定するための情報と、サーバが受信接続を待機する方法についての情報があります。エントリを異なるプラットフォームの異なる Sybase 製品で使用できるようにするには、“ アドレス属性 ” のプロトコル・フィールドとアドレス・フィールド (たとえば、“TCP” と “foobar 5000” など) を、プラットフォームや製品に依存しない形式にする必要があります。

LDAP では各属性の複数のエントリをサポートしているので、各アドレス属性は単一サーバのアドレス (プロトコル、アクセス・タイプ、アドレスを含む) を持つ必要があります。表 5-2 の sybaseAddress を参照してください。

次の例は、異なる接続プロトコルの 2 つのアドレスで受信している NT サーバの LDAP エントリを示します。

```
sybaseAddress = TCP#1#TOEJAM 4444
sybaseAddress = NAMEPIPE#1#¥pipe¥sybase¥query
```

アドレス・フィールドの各エントリは # 文字で区切ります。表 5-2 (23 ページ) はアドレス属性の各フィールドの値の定義を示します。

サーバ・オブジェクトと属性

ディレクトリ・サービスには、Open Client がアクセスするサーバに関する情報が入っていなければなりません。dscp を使用して、*interfaces* を変更し、LDAP サービスにサーバを追加します。

ディレクトリ・サービスはサーバ・エントリをディレクトリ・オブジェクトとして識別します。各ディレクトリ・オブジェクトには、表 5-3 に示すユニークな属性のセットがあります。これらは、Client-Library と Server-Library によって認識されます。

表 5-3: サーバの属性

属性	説明
Server Object Version	オブジェクト定義のバージョンを示す記号整数コード。オブジェクト定義の将来の変更を識別するために、Sybase がこの属性を提供する。
Server Name	サーバの名前を表す文字列値。名前として有効なのは 512 バイト以下の任意の文字列。 サーバ名属性は、ディレクトリ・エントリを見つけるために使用される名前とは異なる。後者はディレクトリ名の構文で表される、ディレクトリ・エントリのフル・パス名である。 混同しないようにするため、システム管理者は名前の属性が部分的にサーバのフル・パス名と一致するようにする (たとえば、属性値をエントリの共通名にする)。
Server Service	サーバが提供するサービスを示す文字列値。サービス値として有効なのは 512 バイト以下の任意の文字列。
Server Status	サーバの動作ステータスを示す記号整数コード。有効な値： 1 － アクティブ 2 － 停止 3 － 失敗 4 － 不明
トランスポート・アドレス	サーバに対する 1 つ以上のトランスポート・アドレス。 トランスポート･アドレス属性には､次の 2 つの要素がある。 • トランスポート・タイプ • トランスポート・アドレス
Security Mechanism	サーバがサポートするセキュリティ・メカニズムを指定するための、オブジェクト識別子 (OID) の文字列。この属性はオプション。省略した場合、Open Server は Open Server が対応するセキュリティ・ドライバを持つ任意のセキュリティ・メカニズムにクライアントが接続できるようにする (詳細については、「Server-Library とセキュリティ・サービス」(36 ページ) を参照。) OID の詳細については、「objectid.dat ファイル」(88 ページ) を参照。例については、*$SYBASE/$SYBASE_OCS/config/objectid.dat* の [SECMECH] セクションを参照。

アプリケーションでのディレクトリ・サービスの使い方

Client-Library と Server-Library は、サーバのアドレスを取得するときに、*interfaces* ファイルではなく、ディレクトリ・サービスを使用できます。

ディレクトリ・サービスから情報を検索するために、Open Client/Open Server ソフトウェアはディレクトリ・ドライバを使用します。ディレクトリ・ドライバは、特定のディレクトリ・サービスに対する汎用インタフェースを Open Client/Open Server ソフトウェアに提供する Sybase ライブラリです。Sybase はサポートするディレクトリ・サービスごとにディレクトリ・ドライバを提供しています。

Client-Library と Server-Library は、次のようにしてディレクトリ・サービスと *interfaces* のどちらを使用するかを決定します。

1 アプリケーションがディレクトリ・ドライバを指定している場合、(Client-Library では ct_con_props (CS_SET, CS_DS_PROVIDER)、Server-Library では srv_props (CS_SET, SRV_S_DSPROVIDER) を呼び出している場合) は、*libtcl.cfg* の DIRECTORY セクションを検証して一致するドライバを探し、そのドライバをロードします。

   ディレクトリ・ドライバと *libtcl*.cfg* の詳細については、「libtcl.cfg ファイルと libtcl64.cfg ファイル」(64 ページ) を参照してください。

2 クライアント・アプリケーションがディレクトリ・ドライバを指定していない場合は、Client-Library と Server-Library は *libtcl.cfg* ファイルの [DIRECTORY] セクション内の最初のエントリにリストされているディレクトリ・ドライバをロードします。

3 次のいずれかが当てはまる場合、Client-Library と Server-Library は *interfaces* ファイルに戻り、そこからサーバのアドレスを取得します。

   - *libtcl.cfg* が存在しない。
   - *libtcl.cfg* の [DIRECTORY] セクションにエントリがない。
   - 指定されたディレクトリ・ドライバのロードに失敗した。
   - CS_IFILE が ct_config によって設定されている場合、*libtcl*.cfg* はコンテキスト・レベルで上書きされる。

libtcl.cfg* ファイルを使用して LDAP サーバ名、ポート番号、DIT ベース、ユーザ名、パスワードを指定し、LDAP サーバへの接続を認証します。

libtcl.cfg* ファイルについて知っていなければならないことは、次のとおりです。

- *libtcl*.cfg* ファイルに指定されている値は、CS_* プロパティのデフォルトになります。これは、ct_con_props() によって設定されます。その特定の接続に ct_con_props() を明示的に設定することで、これらの値を上書きできます。
- *libtcl*.cfg* ファイルのパスワードまたはユーザ名のどちらかを指定しない場合、接続は匿名になります。
- パスワードが 0x で始まっている場合、接続属性ではパスワードは暗号化されていると想定します。詳細については、「パスワードの暗号化」(67 ページ) を参照してください。
- 64 ビットのプラットフォームでは、Open Client/Open Server には 32 ビットと 64 ビットの両方のバイナリがあります。32 ビット・アプリケーションと 64 ビット・アプリケーションの互換性を保つためには、*libtcl.cfg* と *libtcl64.cfg* ファイルの両方を編集してください。

libtcl.cfg* ファイルは、*$SYBASE/$SYBASE_OCS/config* ディレクトリにあります。

接続のプロセスは次の基本手順に従います。

1　Client-Library は *libtcl*.cfg* ファイルに指定されている Sybase ディレクトリ・ドライバを使用して、my_server のアドレスを要求します。

2　ディレクトリ・サービスは my_server エントリの属性を調べて、その情報を Sybase ディレクトリ・ドライバを使用して Client-Library に返します。

3　アプリケーションは、このアドレスを使用して my_server があるマシンに接続します。

アプリケーションでの LDAP ディレクトリ・サービスの使い方

Sybase LDAP の機能を使用するには、ベンダ提供マニュアルに従って、LDAP サーバをインストールして設定します。Sybase では LDAP サーバを提供していません。Sybase では Netscape LDAP SDK クライアント・ライブラリを提供しており、Open Client/Open Server には、LDAP ドライバが含まれています。これは、*$SYBASE/$SYBASE_OCS/lib* にあります。

Netscape LDAP SDK ライブラリのロケーションと環境変数は、表 5-5 (30 ページ) にリストされています。

警告！ Sybase LDAP ディレクトリ・サービスでは、DB-Library で構築されたクライアント・アプリケーションはサポートしていません。

LDAP ドライバが LDAP サーバに接続すると、サーバは、匿名アクセスおよびユーザ名とパスワード認証という 2 つの認証方法をベースとした接続を確立します。

- 匿名アクセス － 認証情報を必要としないので、属性を設定する必要はありません。匿名アクセスは、一般には読み取り専用特権に使用します。
- ユーザ名とパスワード － LDAP URL (「libtcl.cfg ファイルと libtcl64.cfg ファイル」(64 ページ) を参照) の拡張機能として *libtcl.cfg* ファイル (64 ビットのプラットフォームでは、*libtcl64.cfg* ファイル) で指定するか、Client-Library に対するプロパティ呼び出しで設定できます。ctlib を介して LDAP サーバに渡されるユーザ名とパスワードは、Adaptive Server へのログインに使用されるユーザ名とパスワードとは別のものです。Sybase では、ユーザ名とパスワード認証を使用されることを強くおすすめします。

認証

クライアント・アプリケーションは、ホスト名とポート番号または IP アドレスを使用して、LDAP サーバへの接続を作成します。この接続はバインドと呼ばれ、安全でないこともありますが、その場合はユーザ名とパスワードの認証を使用できます。可能なアクセスのタイプは、サーバが決定します。

匿名接続

認証を必要としない接続は、匿名接続と呼ばれます。LDAP と Netscape Directory Services はデフォルトで匿名接続が可能です。

匿名アクセス

- 接続の確立には、パスワードなどの認証情報は必要ありません。
- 接続には、追加属性を設定する必要はありません。
- 一般的に、read-only アクセスです。

ユーザ名とパスワード認証

書き込みを許可するアクセス・パーミッションに対しては、基本的なセキュリティの使用をおすすめします。ユーザ名とパスワードは、LDAP サーバへの接続に対して、基本レベルのセキュリティを提供します。ユーザ名とパスワードは、32 ビット・プラットフォームでは *libtcl.cfg* ファイルに、64 ビット・プラットフォームでは *libtcl64.cfg* ファイルに格納できます。また、Client-Library 属性で設定することもできます。*libtcl*.cfg* ファイルと設定ファイルでのパスワードの暗号化については、「付録 B 設定ファイル」を参照してください。

LDAP ディレクトリ・サービスの有効化

注意 LDAP だけが、リエントラント・ライブラリでサポートされています。LDAP ディレクトリ・サービスを使用してサーバに接続する場合は、isql ではなく、isql_r を使用してください。リエントラント・ドライバを使用するには、*libtcl.cfg* の DRIVERS セクションの変更が必要な場合があります。

❖ ディレクトリ・サービスを使用する

1　ベンダ提供のマニュアルに従って、LDAP サーバを設定します。

2　パス環境変数をユーザ・プラットフォームの LDAP ライブラリに追加します。次に例を示します。

```
setenv LD_LIBRARY_PATH|
${LD_LIBRARY_PATH}:$SYBASE/$SYBASE_OCS/lib3p
```

注意 ユーザ・プラットフォームの環境変数とライブラリについては、表 5-5 (30 ページ) を参照してください。

3　ディレクトリ・サービスを使用するように *libtcl*.cfg* ファイルを設定します。

標準的な ASCII テキスト・エディタを使用して、次のように修正します。

- *libtcl*.cfg* ファイルの *[DIRECTORY]* エントリにある LDAP URL 行の行頭から、コメント・マーカのセミコロン (;) を削除します。
- *[DIRECTORY]* エントリに LDAP URL を追加します。サポートされている LDAP URL 値については、表 5-2 (23 ページ) を参照してください。

警告！ LDAP URL は、1 行で記述する必要があります。

このエントリのコンテキストは次のとおりです。

```
ldap=libdldap.so ldap://host:port/ditbase??scope??
     bindname=username?password
```

次に例を示します。

```
[DIRECTORY]
ldap=libdldap.so ldap://huey:11389/dc=sybase,dc=com??
     one??bindname=cn=Manager,dc=sybase,dc=com?secret
```

“*one*” は、DIT ベースの 1 つ下のレベルのエントリを取り出す検索のスコープを示します。

表 5-4 は、*ldapurl* 変数のキーワードの定義を示します。

表 5-4: ldapurl 変数

キーワード	説明	デフォルト	CS_* property
host (必須)	LDAP サーバを実行しているマシンのホスト名または IP アドレス	なし	
port	LDAP サーバが受信に使用しているポート番号	389	
ditbase (必須)	デフォルトの DIT ベース	なし	CS_DS_DITBASE
username	認証するユーザの DN (識別名)	NULL (匿名認証)	CS_DS_PRINCIPAL
password	認証されるユーザのパスワード	NULL (匿名認証)	CS_DS_PASSWORD

4 正しい環境変数が、必要なサードパーティ・ライブラリを指していることを確認してください。表 5-5 は、Netscape LDAP SDK ライブラリのロケーションのリストです。

表 5-5: 環境変数

プラットフォーム	環境変数	ライブラリのロケーション
HP Tru64 UNIX	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
HP-UX 32 ビット	SHLIB_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
HP-UX 64 ビット	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p64*
Linux	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
IBM RS6000 32 ビット	LIBPATH	*$SYBASE/$SYBASE_OCS/lib3p*
IBM RS6000 64 ビット	サポートされていない	
Sun Solaris 32 ビット	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
Sun Solaris 64 ビット	LD_LIBRARY_PATH_64	*$SYBASE/$SYBASE_OCS/lib3p64*

5 dscp または dsedit を使用して、LDAP サーバにエントリを追加します。詳細については、「サーバ・エントリの追加と変更」(42 ページ) と「ディレクトリ・サービスへのサーバの追加」(53 ページ) を参照してください。

LDAP を使った複数ディレクトリ・サービス

高可用性フェールオーバ保護には、複数のディレクトリ・サービスを指定できます。リストにあるディレクトリ・サービスのすべてが LDAP サーバである必要はありません。次に例を示します。

```
[DIRECTORY]
   ldap=libdldap.so ldap://test:389/dc=sybase,dc=com
   dce=libddce.so ditbase=/.:/subsys/sybase/dataservers
   ldap=libdldap.so ldap://huey:11389/dc=sybase,dc=com
```

この例では、*test:389* への接続が失敗すると、指定された DIT ベースを持つ DCE ドライバへの接続にフェールオーバします。この接続も失敗すると、*huey:11389* 上の LDAP サーバに接続しようとします。ベンダが異なると、DIT ベースのフォーマットも異なります。これらの構造体については、『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。

DCE ディレクトリ・サービスの設定作業

Client-Library アプリケーションと Server-Library アプリケーションがディレクトリ・サービスを使用できるようにするには、*libtcl.cfg* を設定します。

1　*libtcl.cfg* の DIRECTORY セクションにディレクトリ・ドライバと DIT ベースを指定します。

2　DCE ディレクトリ構造に DIT ベースが存在することを確認します。

3　Open Client/Open Server ソフトウェアは、DIRECTORY セクションの最初のエントリをデフォルト・ディレクトリ・ドライバとして使用します。

ディレクトリ・ドライバと *libtcl.cfg* の詳細については、「libtcl.cfg ファイルと libtcl64.cfg ファイル」(64 ページ) を参照してください。

4　次に、該当するユーティリティを使用して、ディレクトリ・サービスにターゲット・サーバ用のエントリを作成して、ディレクトリ・サービスを設定します。

dscp の使い方については、「第 7 章 dscp の使用」を参照してください。

dsedit の使い方については、「第 8 章 dsedit の使用」を参照してください。

第 6 章

セキュリティ・サービスの使い方

Client-Library アプリケーションと Server-Library アプリケーションは、サード・パーティのセキュリティ・ソフトウェアが提供するセキュリティ・サービスを使用して、ユーザを認証し、ネットワーク上のマシン間で送信されるデータを保護することができます。

この章では、ネットワークベースのセキュリティがどのように機能するかと、この機能を使用するにはどのような設定が必要かを説明します。

注意 Client-Library 11.1 以降のネットワークベースのセキュリティには、Open Server 11.1 以降をベースにしたサーバが必要です。Open Client DB-Library と SQL Server 11.0 以前はネットワークベースのセキュリティ・サービスをサポートしていません。

ネットワークベース・セキュリティの概要

分散クライアント／サーバ・コンピューティング環境では、不法侵入者が機密データを見たり操作したりするおそれがあります。ネットワークベースのセキュリティでは、サード・パーティの分散セキュリティ・ソフトウェアを利用して、ユーザを認証し、ネットワーク上のマシン間で送信されるデータを保護します。

セキュリティ・メカニズム

Sybase が定義する「セキュリティ・メカニズム」とは、接続時にセキュリティ・サービスを提供する外部ソフトウェアです。UNIX プラットフォームでは、CyberSafe Challenger が提供する Kerberos セキュリティ・サービスと同様に、Distributed Computing Environment (DCE) が提供するセキュリティ・メカニズムを使用できます。

サーバがサポートするセキュリティ・メカニズムを *interfaces* またはディレクトリ・サービスに指定します。*interfaces* とディレクトリ・サービスの secmech の行／属性の値は、ユーザの *objectid.dat* ファイルの [secmech] セクションで定義されているオブジェクト識別子に関連する文字列と対応していなければなりません。

- *interfaces* エントリのオプションの secmech 行には、サーバがサポートしているセキュリティ・メカニズムを指定します。
- ディレクトリ・サービス・エントリの secmech 属性には、サーバがサポートしているセキュリティ・メカニズムを記述します。

クライアントは、サーバのアドレスを取得するときに、クライアントが使用するセキュリティ・メカニズムをサーバがサポートしていることを確認できます。

- secmech 行または属性が指定されていて、セキュリティ・メカニズムがリストされている場合、それらのセキュリティ・メカニズムだけが使用できます。
- secmech 行または属性が指定されていない場合、すべてのセキュリティ・メカニズムを使用できます。
- secmech 行または属性が指定されていても、セキュリティ・メカニズムがリストされていない場合は、サーバはどのセキュリティ・メカニズムもサポートしません。

セキュリティ・ドライバ

Sybase では、Client-Library および Server-Library とセキュリティ・メカニズムとの通信を可能にするセキュリティ・ドライバを提供しています。Sybase の各セキュリティ・ドライバは汎用インタフェースをセキュリティ・プロバイダのインタフェースにマップします。

接続でセキュリティ・メカニズムを使用するには、次の 2 つの条件がどちらも満たされていなければなりません。

- クライアントとサーバは、互換性のあるセキュリティ・ドライバを使用します。たとえば、DCE ドライバを使用するクライアントには DCE ドライバを使用するサーバが必要です。
- クライアント・アプリケーションは、サーバに接続する前に、接続プロパティを設定することによってサービスを要求します。

セキュリティ・サービス

それぞれのセキュリティ・メカニズムは、クライアントとサーバ間に安全な接続を確立するためのセキュリティ・サービスを提供します。各セキュリティ・サービスは特定のセキュリティ問題に対応しています。

セキュリティ・サービスは大きく 2 つに分けられます。

- 認証サービス
- パケットごとのセキュリティ・サービス

セキュリティ・サービスの詳細については、『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。

Client-Library アプリケーションはセキュリティ・メカニズムのサービスを要求するように接続プロパティを設定します。Open Server アプリケーションはクライアント・スレッドのプロパティを参照して、どのサービスが実行されているかを決定します。

DCE と CyberSafe Challenger が提供するセキュリティ・サービスのリストについては、「付録 D DCE セキュリティ・サービス」と「付録 E CyberSafe Kerberos セキュリティ・サービス」を参照してください。

アプリケーションでのセキュリティ・サービスの使い方

Client-Library アプリケーションと Server-Library アプリケーションはセキュリティ・メカニズムを使用して、認証サービスとパケット単位セキュリティ・サービスを実行できます。セキュリティ・メカニズムは、Client-Library と Server-Library が情報を検証し合う情報交換所のようなものです。

Open Client アプリケーションが認証サービスを要求した場合は、次の処理が行われます。

1　Client-Library はセキュリティ・メカニズムを使用してログインを検証します。セキュリティ・メカニズムはログイン・レコードまたはトークンを返します。セキュリティ・メカニズムは要求されたセキュリティ・サービスに基づいてログイン・トークンを作成します。

2　Client-Library は Open Server アプリケーションとのトランスポート接続を確立し、そのログイン・トークンを送信します。

3　Server-Library はセキュリティ・メカニズムを使用してクライアントのログイン・トークンを認証します。ログインが有効の場合は、Open Server アプリケーションは安全な接続を確立します。

Open Client アプリケーションがパケット単位セキュリティ・サービスを要求した場合は、次の処理が行われます。

1　Client-Library はセキュリティ・メカニズムを使用して、Open Server アプリケーションに送信するデータ・パケットを用意します。セキュリティ・メカニズムは、要求されたセキュリティ・サービスに応じて、データを暗号化するか、データに対応する暗号サインを作成します。

2　Client-Library は Open Server アプリケーションにデータ・パケットを送信します。

3 Open Server は、データ・パケットを受信すると、セキュリティ・メカニズムを使用して必要な暗号解読と検証を行います。

Client-Library のセキュリティ機能の詳細については、『Open Client Client-Library/C リファレンス・マニュアル』の「セキュリティ機能」を参照してください。

Client-Library と セキュリティ・サービス

セキュリティ・メカニズムとセキュリティ・メカニズムのサービスを要求するように、Open Client アプリケーションの接続プロパティを設定できます。Client-Library は、接続に使用するセキュリティ・メカニズムとサービスを次のようにして決定します。

1 クライアント・アプリケーションにセキュリティ・ドライバの名前が指定されている場合は、Client-Library は *libtcl.cfg* の SECURITY セクションを調べて、一致するドライバを探してそのドライバをロードします。

2 クライアント・アプリケーションにセキュリティ・ドライバの名前が指定されていない場合は、Client-Library は *libtcl.cfg* の [SECURITY] セクション内の最初のエントリにリストされているセキュリティ・ドライバをロードします。

3 Client-Library は、クライアント・アプリケーションからの接続に使用されるセキュリティ・サービスを決定します。

libtcl.cfg が存在しない場合や、[SECURITY] セクションにエントリが存在しない場合は、ネットワーク・セキュリティ・プロバイダは存在しません。この場合は、ユーザが正しいパスワードを入力したら、Open Server アプリケーションはユーザを認証します。

Server-Library とセキュリティ・サービス

Open Server アプリケーションはクライアント接続要求のプロパティを参照して、使用するセキュリティ・メカニズムと実行するサービスを決定できます。

デフォルトでは、Open Server アプリケーションは *libtcl.cfg* の [SECURITY] セクションにリストされているセキュリティ・メカニズムをサポートしています。secmech 属性をサーバのディレクトリ・エントリに追加するか、secmech 行を Open Server アプリケーションの *interfaces* エントリに追加することによって、管理者はサポートされているメカニズムのリストをさらに制限できます。

Open Client アプリケーションが Open Server アプリケーションからのセキュリティ・セッションを要求すると、次の処理が行われます。

1 Server-Library は、クライアント接続要求と一緒に送信されたセキュリティ・トークンを読み込みます。セキュリティ・トークンには、クライアントが使用するセキュリティ・メカニズムのオブジェクト識別子が入っています。

2　Open Server アプリケーションの *interfaces* エントリまたはディレクトリ・サービス・エントリに secmech 行／属性がリストされている場合は、Server-Library はこの secmech 行／属性を調べて、セキュリティ・トークンに指定されているオブジェクト識別子に対応する値を探します。対応する値が見つからない場合、接続要求は拒否されます。

3　Server-Library は *objectid.dat* を調べて、セキュリティ・メカニズムのローカル名に対応するオブジェクト識別子を探します。

objectid.dat の詳細については、「付録 B 設定ファイル」を参照してください。

4　Server-Library はセキュリティ・メカニズムのローカル名に対応するセキュリティ・ドライバをロードします。

セキュリティ・ドライバは *libtcl.cfg* の [SECURITY] セクションにリストされています。

設定作業

Open Client/Open Server アプリケーションがセキュリティ・サービスを使用できるようにするには、次の手順に従ってください。

- libtcl.cfg の設定
- DCE または CyberSafe Kerberos の設定

以下の項で、これらの作業についてそれぞれ説明します。

libtcl.cfg の設定

libtcl.cfg の [SECURITY] セクションにセキュリティ・ドライバを指定します。

注意　Open Client/Open Server ソフトウェアは [SECURITY] セクションの最初のエントリをデフォルト・セキュリティ・ドライバとして使用します。

セキュリティ・ドライバと *libtcl.cfg* の詳細については、「付録 B 設定ファイル」を参照してください。

DCE または CyberSafe Kerberos の設定

DCE については、「付録 D DCE セキュリティ・サービス」と DCE のマニュアルを参照してください。CyberSafe については、「付録 E CyberSafe Kerberos セキュリティ・サービス」と CyberSafe のマニュアルを参照してください。

オプションで、サーバがサポートしているセキュリティ・メカニズムを制限するには、次の手順に従ってください。

- アプリケーションが *interfaces* を使用する場合は、dscp を使用して、サーバの *interfaces* エントリに secmech 行を追加します。
- アプリケーションがディレクトリ・サービスを使用する場合は、サーバのディレクトリ・サービスに secmech 属性を追加します。

ディレクトリ・サービスまたは *interfaces* に情報を追加する方法については、「第 7 章 dscp の使用」を参照してください。

第 7 章

dscp の使用

この章では、dscp を使用して *interfaces* ファイルを設定する方法とディレクトリ・サービスを設定する方法について説明します。

dscp について

dscp は、*interfaces* ファイルまたは DCE ディレクトリ・サービスの、それぞれのサーバ・エントリを表示、編集するのに使用するコマンド・ライン・ユーティリティです。セッションをオープンしたあとも、これらのコマンドを使用して、必要に応じて、設定のチェック、既存エントリの表示、新しいエントリの作成、エントリの変更を行うことができます。ユーザのシステムに X-Window がインストールされていない場合は、これらのユーティリティを使用します。

注意 dsedit ユーティリティは、*interfaces* または DCE ディレクトリ・サービスのサーバ・エントリを表示、編集するときに使用する、X-Windows ベースのグラフィカル・ツールです。詳細については、「第 8 章 dsedit の使用」を参照してください。

---

dscp の起動

エントリを追加または変更するには、必要な特権でディレクトリ・サービスにログインしてから、dscp を起動します。

dscp を起動するには、次のように入力します。

```
$SYBASE/$SYBASE_OCS/bin/dscp
```

dscp のプロンプト >> が表示されます。表 7-1 は、使用できるコマンドを示します。

表 7-1: dscp コマンド

コマンド	説明
open [*DSNAME*]	指定のディレクトリ・サービスまたは *interfaces* でセッションをオープンする。 dscp － *interfaces* とのセッションをオープンするには、*DSNAME* に “InterfacesDriver” を指定する。
sess	オープンされているすべてのセッションを表示する。
[switch] SESS	セッション番号 *SESS* を現在のセッションにする。
close [SESS]	*SESS* 番号で示されたセッションをクローズする。*SESS* を指定しなかった場合は、現在のセッションをクローズする。
list [all]	現在のセッションのサーバ・エントリを表示する。 エントリの名前を表示するには、list コマンドを使用する。各エントリの属性もリストするには、list all コマンドを使用する。
read *SERVERNAME*	サーバ・エントリ *SERVERNAME* の内容を画面に表示する。
add *SERVERNAME*	現在のセッションにサーバ・エントリ *SERVERNAME* を追加する。 dscp は、*SERVERNAME* についての情報を要求する。角かっこ ([ ]) 内に表示されているデフォルト値を受け入れる場合には、[Return] を押す。
addattr *SERVERNAME*	現在のセッションのサーバ･エントリ *SERVERNAME* に属性を追加する。
mod *SERVERNAME*	現在のセッションのサーバ・エントリ *SERVERNAME* を変更する。 dscp は、*SERVERNAME* についての情報を要求する。角かっこ ([ ]) 内に表示されているデフォルト値を受け入れる場合には、[Return] を押す。
del *SERVERNAME*	現在のセッションのサーバ・エントリ *SERVERNAME* を削除する。
delete-all	現在のセッションのサーバ・エントリをすべて削除する。
copy *NAME1* to {*NAME2* | *SESS* | *SESS NAME2*}	現在のセッションのサーバ・エントリ *NAME1* を次のロケーションにコピーする。 • 現在のセッションのサーバ・エントリ *NAME2* • セッション *SESS* • セッション *NAME2* のサーバ・エントリ *NAME2*
copyall to *SESS*	現在のセッションのすべてのサーバ・エントリをセッション *SESS* にコピーする。
config	Sybase 環境に関する設定情報を画面に出力する。
exit, quit	dscp を終了する。
help, ?, h	ヘルプ画面を表示する。

設定の表示

config コマンドを使用して、現在の Open Client/Open Server の設定とディレクトリ・サービス・プロバイダ名を表示します。

次のコマンドを入力します。

```
config
```

dscp ユーティリティは次の情報を画面に出力します。

- SYBASE 環境変数の値
- ドライバ設定ファイルのロケーション
- dscp セッションをオープンできるディレクトリ・サービス・プロバイダの名前

ヘルプ情報

dscp のヘルプ画面を表示するには、次のいずれかのコマンドを入力します。

```
help
h
?
```

dscp セッションの使用

サーバ・エントリを表示、追加、変更するには、まず、セッションをオープンしてください。dscp セッションをオープンすると、*interfaces* ファイルと対話できます。

一度に複数のセッションをオープンできます。

セッションのオープン

interfaces のセッションをオープンするには、次のように入力してください。

```
open InterfacesDriver
```

セッションをオープンすると、dscp はセッション番号を通知します。たとえば、open InterfacesDriver コマンドを使用して *interfaces* ファイルとのセッションをオープンすると、dscp は次のメッセージを返します。

```
ok
Session 1 InterfacesDriver>>
```

セッションのリスト

すべてのオープン・セッションをリストするには、次のように入力してください。

```
sess
```

オープン・セッション間の切り替え

別のオープン・セッションに切り替えるには、次のように入力してください。

```
switch SESS
```

SESS はセッション番号です。次に、例を示します。

```
switch 3
```

これでセッション 3 に切り替わります。switch キーワードはオプションです。次のように入力することもできます。

```
3
```

これでもセッション 3 に切り替わります。

セッションのクローズ

セッションをクローズするには、次のように入力してください。

```
close SESS
```

ここでは、*SESS* には、セッション番号が入ります。次に、例を示します。

```
close 3
```

セッション 3 がクローズされます。sess コマンドを使用して、すべてのオープン・セッションをリストします。

SESS を指定しないと、現在のセッションがクローズされます。

サーバ・エントリの追加と変更

ディレクトリ・サービスまたは *interfaces* ファイルとのセッションをオープンしたあと、関連するサーバ・エントリのリスト、追加、変更、削除を行うことができます。

注意 サーバ・エントリを追加または変更すると、dscp は自動的に master 行と query 行を作成または変更します。*interfaces* ファイル・エントリの master 行と query 行には、同じ情報が入っています。

各サーバ・エントリは、一連の属性で構成されます。サーバ・エントリを追加または変更すると、dscp は各属性についての情報を要求します。表 7-2 は、各属性を示します。

表 7-2: サーバの属性

属性	値のタイプ	デフォルト値	サーバ・エントリの追加または変更時に変更可能か
Server Object Version	整数	110	追加 ディレクトリ・サービス：不可 *interfaces*：不可 変更 ディレクトリ・サービス：可能 *interfaces*：不可
Server Name	文字列	該当なし	追加 ディレクトリ・サービス：該当なし *interfaces*：該当なし 変更 ディレクトリ・サービス：不可 *interfaces*: 不可
Server Service	文字列	SQL SERVER	追加 ディレクトリ・サービス：可能 *interfaces*：可能 変更 ディレクトリ・サービス：可能 *interfaces*：不可
Server Status	整数	4 有効な値： 1 － アクティブ 2 － 停止 3 － 失敗 4 － 不明	追加 ディレクトリ・サービス：不可 *interfaces*：不可 変更 ディレクトリ・サービス：可能 *interfaces*：不可
Transport Address • Transport type • Transport address	トランスポート・タイプ： 文字列 トランスポート・アドレス： 文字列	トランスポート・タイプ：tcp. トランスポート・アドレス：なし 有効な値： トランスポート・タイプ：“tcp”、“spx”、“decne,t”、“tli tcp”、“tli spx” トランスポート・アドレス：指定されたトランスポート・タイプによって認識されるフォーマットの文字列	追加または変更 ディレクトリ・サービス： トランスポート・タイプ：可能 トランスポート・アドレス：可能 *interfaces*： トランスポート・タイプ：可能 トランスポート・アドレス：可能
Security Mechanism	文字列 注意：各サーバ・エントリには、最大 20 のセキュリティ・メカニズム文字列を追加できる。	なし 有効な値：ユーザの *objectid.dat* に定義されているオブジェクト識別子に対応する文字列	追加 ディレクトリ・サービス：可能 *interfaces*：可能 変更 ディレクトリ・サービス：可能 *interfaces*：可能

サーバ・エントリのリスト

セッションに対応するサーバ・エントリの名前をリストするには、次のように入力します。

```
list
```

セッションに対応するサーバ・エントリの属性をリストするには、次のように入力します。

```
list all
```

サーバ属性については、表 7-2 を参照してください。

サーバ・エントリの表示

サーバ・エントリの内容を表示するには、次のように入力してください。

```
read SERVERNAME
```

たとえば、次のコマンドを入力します。

```
read myserver
```

次の情報が表示されます。

```
DIT base for object:  interfaces
 Distinguish name: myserver
 Server Version: 1
 Server Name: myserver
 Server Service: SQL Server
 Server Status: 4 (Unknown)
 Server Address:
     Transport Type: tcp
     Transport Addr: victory 1824
     Transport Type: tcp
     Transport Addr: victory 1828
```

上記のサーバ属性については、表 7-2 を参照してください。

サーバ・エントリの追加

サーバ・エントリを追加するには、次のように入力します。

```
add SERVERNAME
```

dscp ユーティリティは、*SERVERNAME* についての情報を要求します。各属性の値を入力するか、または [Return] を押して角かっこ ([ ]) に表示されているデフォルト値を使用します。

たとえば、次のコマンドを入力します。

```
add myserver
```

dscp ユーティリティは次のような情報の入力を要求します。

```
Service: [SQL Server]
 Transport Type: [tcp] tcp
 Transport Address: victory 8001
 Security Mechanism []:
```

追加モードを終了するには、次のコマンドを入力します。

```
#done
```

サーバ・エントリには、関連するトランスポートのタイプとトランスポート・アドレスの組み合わせを 20 個まで指定できます。

上記のサーバ属性については、表 7-2 を参照してください。

❖ サーバ・エントリを LDAP ディレクトリ・サービスに追加する

dscp を使用して LDAP サーバにエントリを作成するには、*$SYBASE/OCS-12_5/config/libtcl.cfg* ファイルを編集し、使用する LDAP サーバを追加して、LDAP を有効にする必要があります。

警告！ LDAP サーバ・エントリの後ろにスペースを入れると、dscp はデフォルトに戻って interfaces ドライバを使用し、LDAP サーバには接続しません。

---

dscp を使用してサーバをディレクトリ・サービスに追加します。

1　次のコマンドを入力して、dscp を起動します。

```
$SYBASE/$SYBASE_OCS/bin/dscp
```

2　サーバ・エントリの表示、追加、または修正を行うには、セッションをオープンします。

dscp セッションをオープンすると、*libtcl*.cfg* にリストされたドライバを持つディレクトリ・サービスと対話できます。セッションをオープンするには、次のコマンドを入力します。

```
open DSNAME
```

DSNAME は、ディレクトリ・サービスの名前です。

DSNAME を指定しない場合は、dscp は *libtcl*.cfg* ファイルで指定されたデフォルトのディレクトリ・サービス・プロバイダを使用します。*libtcl*.cfg* ファイルにエントリがない場合は、dscp は *$SYBASE* にあるデフォルトの *interfaces* ファイルを使用します。

3　LDAP サーバへの接続は、次にようになります。

```
Session 1 ldap>>
```

LDAP サーバがログインにユーザ認証を必要とする場合は、サーバ接続時に -U*username* コマンドライン・パラメータ・フラグを使用してください。

匿名アクセスができるように LDAP サーバが設定されている場合は、ユーザ名とパスワードは不要です。ユーザ名とパスワードが *libtcl*cfg* ファイルに指定されている場合は、dsedit と dscp ユーティリティはこれらの変数を使用します。

4 次のコマンドを入力して、ディレクトリ・サービスにサーバを追加します。

```
add server_name
```

server_name は、追加されるサーバの名前です。

5 次のプロンプトでサービス・タイプを指定します。Adaptive Server は、次のデフォルト値になります。

```
Service [ASE Server]
```

[Enter] を押して、デフォルトを受け入れます。

6 トランスポート・タイプを入力します。[Enter] を押して TCP のデフォルト値を受け入れるか、表 5-3 の値を入力します。

7 トランスポート・アドレスを入力します。有効なエントリは、指定されたトランスポート・タイプを有効にする値です。たとえば、TCP 接続では次のように入力します。

```
host_name port_number.
```

8 LDAP サーバ・エンティティは複数のアドレス・エントリを持つことができるので、もう一度「トランスポート・タイプ」が要求されます。別のトランスポート・タイプを入力するか、フィールドはブランクのまま [Enter] キーを押してこのプロンプトを省略し、次に進みます。

9 プロンプトで、追加のトランスポート・タイプに対応する別の有効なトランスポート・アドレスを入力するか、フィールドは空白のまま [Enter] を押して、次に進みます。

10 オプションで、セキュリティ・メカニズム OID を入力します。

11 オプションで、フェールオーバ用のセカンダリ・サーバを入力します。

12 [Enter] を押します。完了すると、次のメッセージが表示されます。

```
Added server_name done
```

サーバ・エントリを表示するには、次の URL を Netscape に入力します。

```
ldap://host:port/ditbase??one
```

次に例を示します。

```
ldap://huey:11389/dc=sybase,dc=com??one
```

注意 Microsoft Internet Explorer では、LDAP URL は認識されません。

サーバ・エントリの変更

既存のサーバ・エントリを変更するには、次のように入力します。

```
mod SERVERNAME
```

dscp は、*SERVERNAME* についての情報を要求します。各属性の値を入力するか、[Return] を押して角かっこ ([ ]) に表示されている既存の値を使用します。

たとえば、次のコマンドを入力します。

```
mod myserver
```

dscp ユーティリティは次のような情報の入力を要求します。

```
Version: [1]
 Service: [SQL Server] Open Server
 Status: [4]
 Address:
     Transport Type: [tcp]
     Transport Address: [victory 1824] victory 1826
     Transport Type: [tcp]
     Transport Address: [victory 1828]
     Transport Type: []
     Security Mechanism []:
```

注意　dscp はバージョン、サービス、ステータス・エントリを変更できません。

アドレスを削除するには、次のコマンドを入力します。

```
#del
```

変更モードを終了するには、次のコマンドを入力します。

```
#done
```

サーバ・エントリの削除

セッションに関連付けられている 1 つまたはすべてのエントリを削除できます。1 つのエントリを削除するには、次のように入力します。

```
del SERVERNAME
```

たとえば、次のコマンドを入力します。

```
del myserver
```

dscp ユーティリティは “myserver” のエントリを削除します。セッションに関連付けられているすべてのエントリを削除するには、次のように入力します。

```
delete-all
```

サーバ・エントリのコピー

dscp では、1 つのセッション内、または複数のセッション間でサーバ・エントリをコピーできます。これには、*interfaces* からディレクトリ・サービスへのエントリのコピーも含まれます。

サーバ・エントリをコピーする場合は、次の 4 つのオプションがあります。

- サーバ・エントリを現在のセッション内に新しい名前でコピーする。
- サーバ・エントリを異なるセッションにコピーする。
- サーバ・エントリを異なるセッションに新しい名前でコピーする。
- 現在のセッション内のすべてのエントリを異なるセッションにコピーする。

セッション内のエントリのコピー

新しいサーバ・エントリを作成する場合は、セッション内でサーバ・エントリをコピーできます。セッション内でエントリをコピーするには、次のように入力します。

```
copy NAME1 to NAME2
```

たとえば、次のコマンドを入力します。

```
copy myserver to my_server
```

dscp は、“myserver” と同じ新しいエントリ “my_server” を作成します。このようにして、新しいエントリを変更し、元のエントリをそのままにしておくことができます。

セッション間のエントリのコピー

セッション間のサーバ･エントリのコピーには､次の 2 つのタイプがあります。

- 既存のサーバ・エントリの名前をそのまま使用する。
- サーバ・エントリの名前を変更する。

エントリを異なるセッションにコピーして、サーバ名をそのまま使用するには、次のように入力します。

```
copy NAME1 to SESS
```

構文の説明は次のとおりです。

- *NAME1* は現在のサーバ名。
- *SESS* はサーバ・エントリのコピー先セッションの番号。

次に、例を示します。

```
copy myserver to 2
```

dscp は現在のセッションの “myserver” エントリをセッション 2 にコピーします。

エントリを異なるセッションにコピーして、異なる名前を付けるには、次のように入力します。

```
copy NAME1 to SESS NAME2
```

各パラメータの意味は、次のとおりです。

- *NAME1* は現在のサーバ名。
- *SESS* はサーバ・エントリのコピー先セッションの番号。
- *NAME2* は新しいサーバ名。

次に、例を示します。

```
copy myserver to 2 my_server
```

dscp は現在のセッションの “myserver” エントリをセッション 2 にコピーして、名前を “my_server” に変更します。

すべてのエントリを別のセッションにコピーする

現在のセッションにあるすべてのエントリを別のセッションにコピーするには、次のように入力します。

```
copyall SESS
```

SESS は全エントリのコピー先セッションの番号です。

たとえば、次のコマンドを入力します。

```
copyall 2
```

dscp は現在のセッションの全エントリをセッション 2 にコピーします。

dscp の終了

dscp を終了するには、次のいずれかのコマンドを入力します。

```
exit
quit
```

第 8 章

dsedit の使用

この章では、dsedit を使用して *interfaces* ファイルを設定する方法と、ディレクトリ・サービスの Sybase サーバのリストを設定する方法について説明します。

dsedit について

dsedit は、*interfaces* ファイルまたは DCE ディレクトリ・サービスのサーバ・エントリを表示、編集するのに使用する X-Windows ベースのグラフィカル・ツールです。

使用しているシステムが X-Windows をサポートしていない場合は、*interfaces* または DCE ディレクトリのサーバ・エントリの設定には dscp を使用します。詳細については、「第 7 章 dscp の使用」を参照してください。

dsedit の起動

サーバを追加または変更する場合は、*interfaces* または DCE ディレクトリを編集できるかどうかを確認してから、dsedit を起動します。

- *interfaces* エントリを編集するには、*interfaces* ファイルに対する**書き込み**パーミッションが必要です。
- DCE ディレクトリ・エントリを編集するには、管理者の特権を持つユーザとして DCE にログインする必要があります。

dsedit を起動するには、次のように入力します。

```
$SYBASE/$SYBASE_OCS/bin/dsedit
```

リモート・マシンから dsedit を実行する場合は、DISPLAY 環境変数が正しく設定されているかどうかを確認してください。DISPLAY 環境変数の設定方法については、使用している X11 のマニュアルを参照してください。

注意 任意の画面でヘルプ情報を参照するには、[HELP] をクリックします。

セッションのオープン

dsedit を起動すると、まず、メイン画面が表示されます。この画面から、*interfaces* ファイルまたは DCE ディレクトリの編集セッションを選択してオープンします。

interfaces ファイル・セッション

デフォルトの *interfaces* をオープンして編集するには、Sybase *interfaces* ファイルを選択して、[OK] をクリックします。代替ファイルをオープンするには、表示されているファイル名を編集してから、[OK] をクリックします。異なるファイルで複数の *interfaces* ファイル・セッションをオープンできます。

interfaces ファイル・セッションのセッション・ウィンドウには、*interfaces* ファイルのフル・パス名が表示され、*interfaces* ファイルに含まれているサーバ・エントリがリストされます。エントリの追加、変更、コピー、削除を行うには、リストの右側にあるボタンを使用します。

- [Add new server entry] － [Server Entry Editor] ウィンドウが表示されます。このウィンドウで、新しいサーバ・エントリの名前とネットワーク・アドレスを指定します。詳細については、「サーバ・エントリの追加、表示、編集」(56 ページ) を参照してください。

- [Modify server entry] － 選択されているサーバ・エントリについて、ネットワーク・アドレスの表示と変更ができます。リストでサーバを選択してから、[Modify server entry] をクリックします。[Server Entry Editor] ウィンドウに、そのサーバの属性が表示されます。詳細については、「サーバ・エントリの追加、表示、編集」(56 ページ) を参照してください。
- [Copy server entry] － このボタンをクリックすると、1 つ以上のエントリを DCE ディレクトリまたは別の *interfaces* ファイルにコピーできます。サーバ・エントリをコピーする前に、次の手順に従って、サーバ・リストからコピーするエントリを選択してください。
  - エントリを 1 つだけコピーするには、そのエントリを 1 回だけクリックします。
  - 連続する複数のエントリをコピーするには、[Shift] キーを押したまま範囲の最初 (または最後) のエントリをクリックし、最後 (または最初) のエントリをクリックします。
  - 連続していない複数のエントリを選択するには、[Ctrl] キーを押しながら、対象となる各エントリをクリックして選択します。

  コピーするエントリを選択したら、[Copy server entry] をクリックします。新しいウィンドウが開き、変換先ディレクトリ・サービスの選択を要求します。次のように、別の *interfaces* ファイルにコピーできます。
  - エントリを別の *interfaces* ファイルにコピーするには、リストから [Sybase Interfaces File] を選択して、表示されたファイル名を編集し、[OK] をクリックします。
  - DCE ディレクトリにエントリをコピーするには、該当するディレクトリ・サービスを選択して、[OK] をクリックします。

[Close Session] をクリックすると、セッション・ウィンドウがクローズされ、変更が *interfaces* に書き込まれます。

注意　*interfaces* セッション・ウィンドウをいったんクローズして、編集内容を *interfaces* に適用する必要があります。

ディレクトリ・サービスへのサーバの追加

警告！　ほとんどの LDAP サーバには、ディレクトリ・エントリを追加するための ldapadd ユーティリティがありますが、Sybase では、汎用ツールにはないセマンティック・チェックが組み込まれている dscp または dsedit の使用をおすすめします。

各サーバ・エントリは、一連の属性で構成されます。サーバ・エントリを追加または変更すると、dscp から、サーバ属性に関する情報の入力を指示するプロンプト画面が表示されます。属性のいくつかはデフォルトで提供されますが、そのほかはユーザが入力する必要があります。dscp を使用してディレクトリ・サービスを作成すると、角カッコ "[ ]" 内にデフォルト値が表示されます。入力可能な値のリストについては、表 5-2 (23 ページ) を参照してください。

dsedit を使用して、*libtcl*.cfg* と *interfaces* (Windows NT では *sql.ini*) ファイルでのサーバの追加、削除、変更を行うことができます。ただし、LDAP URL を *libtcl*.cfg* ファイルに追加してから、LDAP サーバ・エントリの追加、削除、変更を行ってください。詳細については、「libtcl.cfg ファイルと libtcl64.cfg ファイル」(64 ページ) を参照してください。

❖ dsedit を使用してディレクトリ・サービスにサーバを追加する

1 *$SYBASE/$SYBASE_OCS/bin* ディレクトリから、次のように入力します。

```
dsedit
```

2 サーバの一覧から [LDAP] を選択して、[OK] をクリックします。

3 [Add New Server Entry] をクリックします。

4 次のように入力します。

- サーバ名 – 必須です。
- セキュリティ・メカニズム – オプションです。セキュリティ・メカニズムの OID のリストは、*$SYBASE/$SYBASE_OCS/config/objectid.dat* にあります。
- HA サーバ名 – オプションです。高可用性フェールオーバ・サーバを使用している場合は、その名前を入力します。

5 [Add New Network Transport] をクリックします。

- ドロップダウン・リストからトランスポート・タイプを選択します。
- ホスト名を入力します。
- ポート番号を入力します。

6 [OK] を 2 回クリックして、dsedit ユーティリティを終了します。

サーバ・エントリを表示するには、次の URL を Netscape に入力します。

```
ldap://host:port/ditbase??one
```

次に例を示します。

```
ldap://huey:11389/dc=sybase,dc=com??one
```

注意 Microsoft Internet Explorer では、LDAP URL は認識されません。

DCE ディレクトリ・セッションの使用

セッションをオープンするには、リストからディレクトリ・サービス名を選択し、[OK] をクリックします。*interfaces* 以外のディレクトリ・サービス・エントリが表示されない場合は、*libtcl.cfg* に DCE ディレクトリ・ドライバを設定する必要があります。

DCE ディレクトリ・セッションのセッション・ウィンドウには、エントリが保管される DCE ディレクトリの DIT ベースが表示されます。エントリの追加、変更、コピー、削除を行うには、リストの左側にあるボタンを使用します。

- [Add new server entry] － [Server Entry Editor] ウィンドウが表示されます。このウィンドウで、新しいサーバ・エントリの名前とネットワーク・アドレスを指定します。詳細については、「サーバ・エントリの追加、表示、編集」(56 ページ) を参照してください。
- [Modify server entry] － 選択されているサーバ・エントリについて、ネットワーク・アドレスの表示と変更ができます。[Modify server entry] をクリックし、サーバ名を入力します。詳細については、「サーバ・エントリの追加、表示、編集」(56 ページ) を参照してください。
- [Copy server entry] － このボタンをクリックすると、1 つ以上のエントリを DCE ディレクトリまたは別の *interfaces* ファイルにコピーできます。[Copy server entry] をクリックし、コピーするエントリの名前を入力します。新しいウィンドウが開き、変換先ディレクトリ・サービスの入力を要求します。
  - エントリを別の *interfaces* ファイルにコピーするには、リストから [Sybase Interfaces File] を選択し、表示されたファイル名を編集して、[OK] をクリックします。
  - エントリを別の DCE ディレクトリにコピーするには、該当するディレクトリ・サービスを選択し、[OK] をクリックします (異なる DCE ディレクトリを使用可能なディレクトリ・サービスとして表示するには、*libtcl.cfg* ファイルに、異なる DIT ベース値を指定した複数の DCE ディレクトリ・ドライバ・エントリが必要です。ディレクトリ・ドライバ・エントリの設定方法については、「ディレクトリ・ドライバの追加」(70 ページ) を参照してください)。

[Close session] をクリックすると、セッション・ウィンドウがクローズされます。

サーバ・エントリの追加、表示、編集

[Server Entry Editor] ウィンドウを使用して、*interfaces* ファイルまたは DCE ディレクトリのサーバ・エントリを表示または編集します。[Session] ウィンドウで [Add New Server Entry] ボタンまたは [Modify Server Entry] ボタンをクリックすると、[Server Entry Editor] ウィンドウとそのフィールドが表示されます。

- サーバ名 – サーバ・エントリを追加するには、新しいサーバの名前を入力します。サーバ・エントリを編集する場合は、名前フィールドを編集して、サーバの名前を変更できます (新しい名前は、DCE ディレクトリまたは *interfaces* ファイルにないものを指定してください)。
- 使用可能なネットワーク・トランスポート – サーバがクライアント接続を受け付けるネットワーク・アドレスのリスト。次の手順に従って、このアドレス・リストを編集できます。
  - [Add Network Transport] または [Modify Network Transport] を選択して、新しいアドレスを作成するか、既存のアドレスを編集します。詳細については、次の「ネットワーク・トランスポート・アドレスの追加または編集」を参照してください。
  - [Delete Network Transport] をクリックすると、選択したネットワーク・アドレスが削除されます。
  - サーバ・エントリに複数のアドレスがある場合は、[Move network transport up] または [Move network transport down] をクリックして、リスト内のアドレスの順序を並べ換えることができます。
- [OK] ボタン – 変更を確認してウィンドウをクローズします。*interfaces* に対する変更は、セッションをクローズしないと適用されないことに注意してください。
- [Cancel] ボタン – ウィンドウをクローズし、すべての編集内容を廃棄します。

ネットワーク・トランスポート・アドレスの追加または編集

[Network Transport Editor] では、サーバがクライアント接続を受け付けるトランスポート・アドレスを表示、編集、作成することができます。このウィンドウには、アドレスに対応するサーバ・エントリの名前が表示され、次の項目を設定できます。

- [Transport type] メニュー – アドレスのプロトコルとインタフェースを指定します。有効な値は tcp、tli tcp、tli spx、spx のいずれかです。
- アドレス情報 – トランスポートのタイプによって、必要なアドレスのコンポーネントが異なります。次に、アドレス・フォーマットについて、詳しく説明します。

TCP/IP アドレス

[Transport type] メニューから [tcp] または [tli tcp] を選択すると、TCP/IP アドレスが表示されます。*interfaces* エントリでは、次の場合に "tli tcp" プロトコルを使用してください。

- "tli" フォーマットの *interfaces* エントリを使用するプラットフォームで稼働する、リリース 11.0.x 以前の SQL Server または Replication Server®。
- Solaris で、DB-Library が "tcp" と "tli" の両フォーマットをサポートする場合。

他のクライアントとサーバには、"tcp" トランスポート・タイプを使用します。

DCE ディレクトリ・エントリで "tli tcp" トランスポートを使用することはおすすめできません。DCE ディレクトリは、複数プラットフォームで実行するアプリケーション間で共有できますが、"tli tcp" プロトコルはすべてのプラットフォームで認識されるわけではないからです。DCE ディレクトリ・エントリの TCP/IP アドレスには tcp を使用してください。

TCP/IP エントリのアドレス情報は、ホスト名 (または IP アドレス) とポート番号 (10 進数として入力) で構成されます。"tli tcp" フォーマットの *interfaces* エントリでは、ホストの IP アドレスとポート番号は、"tli tcp" フォーマットの *interfaces* エントリに必要な 16 バイトの 16 進表現に変換されます。

SPX/IPX アドレス

SPX/IPX アドレスの場合は、UNIX Adaptive Server は Novell ネットワークで実行しているクライアント・アプリケーションから接続を受信できます。[Transport type] メニューから “tli spx” または “spx” を選択して、SPX/IPX アドレスを表示します。SPX/IPX は、次の情報で構成されます。

- ホスト・アドレス － 8 桁の 16 進数で、サーバが稼働するコンピュータの IP アドレスを表します。ドットで区切られた 10 進の IP アドレス・フォーマットの各コンポーネントは、16 進アドレス・フォーマットの 1 バイトにマップします。たとえば、IP アドレスが 128.15.15.14 であれば、SPX/IPX のホスト・アドレス値として 800F0F0E と入力します。
- ポート番号 － ポート番号は、4 桁の 16 進数として表されます。
- 終了ポイント － SPX デバイス・ドライバを指すデバイス・ファイルのパス。デフォルトは、Soralis では */dev/mspx*、それ以外のプラットフォームでは */dev/nspx* です。必要に応じ、サーバが稼働するマシンに合わせてパスを変更してください。デフォルトのパスは、dsedit を実行しているプラットフォームをもとにしています。

dsedit または dsedit 問題のトラブルシューティング

ここでは、一般的な問題をいくつか取り上げて、それらの問題を修正する方法について説明します。

dsedit が起動しない

次の各項に該当していないか確認してください。

- SYBASE 環境変数が設定されていない、または間違ったディレクトリを指している。
- X11 が正しく設定されていない。リモート・ホストで dsedit を実行している場合は、リモート・ホストの X11 クライアントがユーザ自身のマシンの X11 サーバに接続できるかどうかを確認してください。トラブルシューティングの詳細については、使用している X11 のマニュアルを参照してください。X11 が使用可能ではない場合は、dsedit の代わりに、dscp または dscp_dce を使用します。
- dsedit を実行しようとしたが、必要な DCE ライブラリがロードできなかった。dsedit には、DCE インストールで提供される共有ライブラリが必要です。使用システムの共有ライブラリ検索パスに、DCE ライブラリが入っているディレクトリがあることを確認してください。

DCE が使用可能なディレクトリ・サービスとして表示されない

次の各項に該当していないか確認してください。

- dsedit を使用して DCE エントリを編集できる。
- *libtcl.cfg* ファイルの [DIRECTORY] セクションが正しく設定されていない。このセクションの各エントリには、[Select Directory Service] メニューに表示されるディレクトリ・サービスのディレクトリ・ドライバが設定されています。ディレクトリ・サービスのドライバの設定方法については、「ディレクトリ・ドライバの追加」(70 ページ) を参照してください。

DCE ディレクトリ・セッションをオープンできない

DCE サービスが [Select Directory Service] メニューに表示されているのに、オープンしようとするとエラーになる場合には、次の点を確認してください。

- *libtcl.cfg* の [DIRECTORY] セクションのエントリに、間違った Sybase ドライバがリストされていることがあります。*libtcl.cfg* ファイル構文の詳細については、「DIRECTORY セクション」(65 ページ) を、ドライバをロードする方法については、「ドライバの動的リンク」(64 ページ) をそれぞれ参照してください。

サーバ・エントリを追加、変更、または削除できない

次の各項に該当していないか確認してください。

- *interfaces* ファイルまたは DCE ディレクトリにパーミッション問題がある。

  interfaces ファイルを編集するには、*interfaces* ファイルと Sybase インストール・ディレクトリに対して書き込みパーミッションが必要です。また、DCE エントリを編集するには、DCE ディレクトリに対する書き込み特権を持っているユーザとして DCE にログインしてください。

- *libtcl.cfg* ファイルで DIT ベースが DCE 向けに誤って設定されている。

  libtcl.cfg [DIRECTORY] セクションの各ディレクトリ・サービス・エントリには、そのサービスのサーバ・オブジェクトが保管されるベース・パスである DIT ベースがリストされています。DIT ベース・パスが DCE ディレクトリに実際に存在することを確認してください。DIT ベースを定義する方法については、「DIRECTORY セクション」(65 ページ) を参照してください。

- DCE システム問題

  DCE 管理者は、DCE dcecp ツールを使用して、DCE ディレクトリ・ソフトウェアが正しく機能していることを検証できます。詳細については、DCE のマニュアルを参照してください。

付 録　A

環境変数

この章では、設定情報となる環境変数を説明します。

接続に使用する環境変数

Open Client/Open Server 製品は、接続処理時に表 A-1 の環境変数を使用します。

表 A-1: 接続に使用する環境変数

変数	値	使用箇所
DSLISTEN	*interfaces* またはディレクトリ・サービスにリストされている Open Server アプリケーションの名前。 DSLISTEN が設定されていない場合、Open Server はデフォルト値 “SYBASE” を使用する。	Open Server
DSQUERY	*interfaces* またはディレクトリ・サービスにリストされているターゲット・サーバの名前。 DSQUERY が設定されていない場合、Open Client はデフォルト値 “SYBASE” を使用する。	Open Client
SYBASE_OCS	Open Client/Open Server 製品のホーム・ディレクトリ。	*$SYBASE/$SYBASE_OCS*

ローカライゼーションで使用する環境変数

注意 *LC_xxxx* 変数は DB-Library では使用されません。

Open Client/Open Server 製品はローカライゼーション時にこれらの環境変数を使用します。

- LC_ALL
- LC_COLLATE
- LC_TYPE
- LC_MESSAGE
- LC_TIME

ローカライゼーション環境変数は、POSIX 標準環境変数であり、Sybase 以外のアプリケーションでも使用可能です。

Sybase 以外のアプリケーションの中には、Open Client/Open Server アプリケーションと同じローカライゼーション関連の環境変数を使用できるものもあります。*locales.dat* には、Sybase 以外のアプリケーションの環境変数で使用するのと同じロケール名をリストするようにしてください。

環境変数の設定

ここでは、C シェル と Bourne シェルで環境変数を設定する手順を説明します。

C シェルで環境変数を設定するには、次のコマンドを使用します。

```
setenv VARIABLE value
```

たとえば、次のコマンドは DSQUERY 環境変数を “test” と定義します。

```
setenv DSQUERY test
```

Bourne シェルで環境変数を設定するには、次のコマンドを使用します。

```
VARIABLE=value
```

たとえば、次のコマンドは DSQUERY 環境変数を “test” と定義します。

```
DSQUERY=test
```

付 録　B

設定ファイル

この章では、Open Client/Open Server 製品が設定情報を入手するときに使用するファイルについて説明します。この項は、次の項目で構成されています。

設定ファイルについて

設定ファイルは、インストール時に *$SYBASE* ディレクトリ構造内のデフォルト・ロケーションに作成されます。Open Client/Open Server 製品は表 B-1 にリストされている設定ファイルを使用します。

表 B-1: 設定ファイルの名前とロケーション

ファイル名	説明	ロケーション	関連項目
libtcl.cfg	このドライバ設定ファイルには、ディレクトリ、セキュリティ、ネットワークの各ドライバに関する情報と、必要な初期化情報が格納されている。	*$SYBASE/$SYBASE_OCS/config*	「libtcl.cfg ファイルと libtcl64.cfg ファイル」(64 ページ) 参照。
interfaces	*interfaces* ファイルには、ファイルにリストされている各サーバの接続とセキュリティ情報が含まれる。このファイルは *libtcl.cfg* ファイルのバックアップとしても使用される。	*$SYBASE*	「interfaces ファイル」(73 ページ) 参照。
objectid.dat	オブジェクト識別子ファイルは、文字セット、照合順、セキュリティ・メカニズムのロケール名にグローバル・オブジェクト識別子をマップする。	*$SYBASE/$SYBASE_OCS/config*	「付録 C ローカライゼーション」参照。
ocs.cfg	ランタイム設定ファイルを使用すると、実行時に特定の値を変更できる。	*$SYBASE/$SYBASE_OCS/config*	「ocs.cfg ファイル」(77 ページ) 参照。

libtcl.cfg ファイルと *libtcl64.cfg* ファイル

libtcl.cfg ファイルと *libtcl64.cfg* ファイル (*libtcl*.cfg* ファイル) は、Open Client/Open Server 製品で使用する 3 つのタイプのドライバ情報を含むドライバ設定ファイルです。

- ディレクトリ・ドライバ
- セキュリティ・ドライバ
- ネットワーク (Net-Library) ドライバ

ドライバは、Open Client/Open Server ソフトウェアに外部サービス・プロバイダとの汎用インタフェースを提供する Sybase ライブラリです。これによって、Open Client/Open Server は、複数のサービス・プロバイダをサポートできます。たとえば、Open Client は DCE ディレクトリ・ドライバを使用して、DCE ディレクトリ・サービスと通信できます。

libtcl.cfg* ファイルの目的は、設定情報 (Open Client/Open Server と Open Client/Open Server ベースのアプリケーション用のドライバ、ディレクトリ、セキュリティ・サービスなど) を提供することです。*libtcl.cfg* と *libtcl64.cfg* は両方とも、64 ビット・プラットフォーム上で提供されます。設定情報を探す場合は、dsedit や srvbuild などの (64 ビット・プラットフォームの) 32 ビット・アプリケーションでは *libtcl.cfg* ファイル、64 ビット・アプリケーションでは *libtcl64.cfg* ファイルを検索します。

libtcl.cfg* ファイルには、*interfaces* ファイルや LDAP ディレクトリ・サービスを使用するかどうかを指定します。*libtcl*.cfg* ファイルに LDAP が指定してある場合は、サーバ接続時に -I パラメータを渡すことによってアプリケーションが明示的に *libtcl*.cfg* ファイルを上書きしないかぎり、*interfaces* ファイルは無視されます。

ドライバの動的リンク

Client-Library と Server-Library は、ディレクトリとセキュリティ・ドライバの動的ロードをサポートしています。これによって、アプリケーションを再リンクすることなく、アプリケーションが使用しているドライバを変更でき、自分のサイトで使用できるようになったときにその機能を使用できます。

$SYBASE/$SYBASE_OCS/config/libtcl.cfg には、Net-Library、ディレクトリ、セキュリティ・ドライバを設定します。このファイルは、記号文字列を適切なドライバと必要な初期化情報にマップします。

dscp_dce などの Sybase ユーティリティ・プログラムを含む Client-Library または Server-Library アプリケーションは、次のように *libtcl.cfg* で指定された適切なドライバを検索します。

1 *libtcl.cfg* 内のドライバのファイル名にパスのコンポーネント (スラッシュを含んでいる) が指定されている場合には、そのパスが使用されます。指定されていない場合は、検索は手順 2 に進みます。

2　ユーザのプラットフォームによっては、次の環境変数によって指定されたディレクトリを検索します。

- Sun Solaris、HP Tru64 UNIX、Linux － LD_LIBRARY_PATH
- HP/UX － SHLIB_PATH
- IBM RS6000 － LIBPATH

ドライバが見つからない場合には、手順 3 に進みます。

3　パス *$SYBASE/$SYBASE_OCS/lib* (または、デバッグモード・ライブラリを使用して構築されたアプリケーションには *$SYBASE/$SYBASE_OCS/devlib*) を使用します。

libtcl.cfg の使い方

ネットワーク、ディレクトリ、またはセキュリティ・ドライバをロードすると、Open Client/Open Server は *libtcl.cfg* ファイルを読み込みます。*libtcl.cfg* は、*$SYBASE/$SYBASE_OCS/config* ディレクトリにあります。

libtcl.cfg のエントリは、Open Client/Open Server 製品にドライバの名前とそのドライバの初期化情報を提供します。

libtcl.cfg の構成

Open Client/Open Server バージョン 12.5.1 のディレクトリ・サービスやセキュリティ・サービスを利用するには、これらのサービスをサポートする適切なソフトウェアが必要です。

libtcl.cfg ファイルは、ドライバのタイプごとに 3 つのセクションに分かれています。セクションには、次のような見出しが付けられています。

- [DIRECTORY]
- [SECURITY]
- [DRIVERS]

DIRECTORY セクション

DIRECTRY セクションには、ディレクトリ・ドライバがリストされています。ディレクトリ・ドライバ・エントリの構文は、次のとおりです。

```
provider=driver init-string
```

各パラメータの意味は、次のとおりです。

- *provider* には、ディレクトリ・サービスのローカル名が入ります。この要素には、アルファベット、数字、アンダースコアだけで構成される、64 文字以内の任意の名前を付けることができます。

- *driver* はドライバの名前です。すべてのドライバのデフォルト・ロケーションは *$SYBASE/$SYBASE_OCS/lib* です。DCE ディレクトリ・ドライバはプラットフォームに依存します。
  - HP/UX − *libddce.sl*
  - IBM RS/6000 − *libddce.so.a*
  - Sun Solaris、HP Tru64 UNIX − *libddce.so*
- *init-string* はドライバの初期化文字列です。*init-string* の値はドライバによって異なります。

DCE ドライバでは、*init-string* には DIT ベースを指定します。DIT ベースは、DCE がサーバ・エントリの検索を開始するロケーションです。*init-string* の構文を次に示します。

ditbase=/.:/*location*

init-string が指定されていない場合は、デフォルトの DIT ベースは */.:/subsys/sybase/dataservers* になります。

DCE dcecp ユーティリティを実行し、create directory コマンドを使用してサーバ・エントリがある可能性のある DCE ディレクトリ構造に DIT ベースのロケーションを作成します。

次の DIRECTORY セクションには、Sun Solaris DCE ドライバのエントリが示されています。

```
[DIRECTORY]
 dce=libddce.so ditbase=/.:/subsys/sybase/server
```

DCE ディレクトリ・ドライバのグローバル・ディレクトリ・サービスとの対話

HP Tru64 UNIX では、DCE のディレクトリ・ドライバがグローバル・ディレクトリ・サービス (GDS) と対話します。グローバル・ディレクトリ・サービスやセル・ディレクトリ・サービスに使用される名前の有効な文字については、DCE のマニュアルを参照してください。

DIRECTORY セクションの LDAP エントリ

最も簡単なフォームで、LDAP ディレクトリ・サービスは、次のようなフォーマットによって指定されています。

```
[DIRECTORY]
ldap=libdldap.so ldapurl
```

ここでは、*ldapurl* は次のように定義されています。

```
ldap://host:port/ditbase
```

次の LDAP エントリは上記と同じ属性を使用していますが、匿名接続であり、LDAP サーバが読み込み専用アクセス可能な場合にだけ動作します。

```
ldap=libdldap.so ldap://test:389/dc=sybase,dc=com
```

libtcl.cfg* ファイルでユーザ名とパスワードを LDAP URL への拡張機能として指定すると、接続時にパスワード認証が有効になります。

ユーザ名を設定するには、次のように入力します。

```
if (ct_con_props(conn, CS_SET, CS_DS_PRINCIPAL, ldapprincipal,
     strlen(ldapprincipal), (CS_INT *)NULL) != CS_SUCCEED)
{
     ...
}
```

パスワードを設定するには、次のように入力します。

```
if (ct_con_props(conn, CS_SET, CS_DS_PASSWORD, ldappassword,
     strlen(ldappassword), (CS_INT *)NULL) != CS_SUCCEED)
{
     ...
}
```

パスワードの暗号化

libtcl.cfg と *libtcl64.cfg* ファイルのエントリは人間が判読できるフォーマットです。Sybase では、基本的なパスワードの暗号化のために pwdcrypt ユーティリティを提供しています。pwdcrypt は、キーボード入力を行うと、パスワードと置換される暗号値を生成する単純なアルゴリズムです。pwdcrypt ユーティリティは *$SYBASE/$SYBASE_OCS/bin* にあります。

Open Client/Open Server (OCS) ディレクトリから、コマンド・プロンプトに次のように入力します。

```
bin/pwdcrypt
```

要求されたら、パスワードを 2 度入力します。

pwdcrypt ユーティリティが、次のように暗号化されたパスワードを生成します。

```
0x01312a775ab9d5c71f99f05f7712d2cded2i8d0ae1ce78868d0e8669313d1bc4c706
```

標準的な ASCII テキスト・エディタを使用して、暗号化されたパスワードをコピーして *libtcl*.cfg* ファイルに貼り付けます。暗号化の前に、ファイル・エントリが次のように表示されます。

```
ldap=libdldap.so
ldap://dolly/dc=sybase,dc=com????bindname=cn=Manager,dc=sybase,dc=com?secret
```

パスワードを、暗号化した文字列に置き換えます。

```
ldap=libdldap.so
ldap://dolly/dc=sybase,dc=com????bindname=cn=Manager,dc=sybase,dc=com?
0x01312a775ab9d5c71f99f05f7712d2cded2i8d0ae1ce78868d0e8669313d1bc4c706
```

警告！ パスワードが暗号化された場合でも、ファイル・システム・セキュリティを使用してパスワードを保護してください。

SECURITY セクション

SECURITY セクションには、セキュリティ・ドライバがリストされています。セキュリティ・ドライバ・エントリの構文は次のとおりです。

```
provider=driver init-string
```

各パラメータの意味は、次のとおりです。

- *provider* には、セキュリティ・メカニズムのローカル名が入ります。セキュリティ・メカニズムのローカル名は、オブジェクト識別子ファイル *$SYBASE/$SYBASE_OCS/config/objectid.dat* にリストされています。

  objectid.dat の詳細については、「objectid.dat ファイル」(88 ページ) を参照してください。

  DCE セキュリティ・メカニズムのデフォルト・ローカル名は “dce” です。CyberSafe Kerberos セキュリティ・メカニズムのデフォルト・ローカル名は “csfkrb5” です。メカニズムのローカル名にデフォルト以外の名前を使用する場合は、オブジェクト識別子ファイル内のデフォルト名の後に、その名前のエイリアスを追加する必要があります (例については、「objectid.dat の例」(89 ページ) を参照してください)。

- *driver* はドライバの名前です。すべてのドライバのデフォルト・ロケーションは *$SYBASE/$SYBASE_OCS/lib* です。

表 B-2 は、プラットフォームごとにサポートされているセキュリティ・ドライバのリストです。

表 B-2: サポートされているセキュリティ・ドライバ

プラットフォーム	セキュリティ・タイプ	セキュリティ・ドライバ	サービスの互換性
Solaris 2.x	DCE	*libsdce.so*	オペレーティングシステム・プロバイダは、OSF DCE 1.1 と互換性がなければならない。
	CyberSafe Kerberos	*libskrb.so*	CyberSafe TrustBroker 2.1。
HP Tru64 UNIX	DCE	*libsdce.so*	オペレーティングシステム・プロバイダは、OSF DCE 1.1 と互換性がなければならない。
IBM RS/6000	DCE	*libsdce.so.a*	オペレーティングシステム・プロバイダは、OSF DCE 1.1 と互換性がなければならない。
	CyberSafe Kerberos	*libskrb.so.a*	
HP-UX	DCE	*libsdce.sl*	オペレーティングシステム・プロバイダは、OSF DCE 1.1 と互換性がなければならない。
	CyberSafe Kerberos	*libskrb.sl*	CyberSafe TrustBroker 2.1。

- *init-string* はドライバの初期化文字列です。値はドライバによって異なります。

  DCE ドライバでは、*init-string* の構文は次のとおりです。

  ```
  secbase=/.../cell_name
  ```

  cell_name は、DCE セルの名前です。

  CyberSafe Kerberos ドライバでは、*init-string* の構文は次のとおりです。

  ```
  secbase=@realm
  ```

  realm は、デフォルトの CyberSafe Kerberos レルム名です。

次の SECURITY セクションには、Sun Solaris 上の DCE と CyberSafe Kerberos ドライバのエントリが示されています。

- DCE

  ```
  [SECURITY]
   dce=libsdce.so secbase=/.../dsatestcell
  ```

- CyberSafe Kerberos

  ```
  [SECURITY]
   csfskrb=libskrb.so secbase=@ASE libgss=libgss library>
  ```

  libgss=/krb5/lib/libgss.so です。

DRIVERS セクション

DRIVERS セクションには、ネットワーク・ドライバがリストされています。ディレクトリ・ドライバ・エントリの構文は、次のとおりです。

```
driver=protocol description
```

各パラメータの意味は、次のとおりです。

- *driver* には、ドライバの名前が入ります。すべてのドライバのデフォルト・ロケーションは *$SYBASE/$SYBASE_OCS/lib* です。

  ネットワーク・トランスポート・ドライバは、次のとおりです。

  - tli 用の *libtli.so* と *libinsck.so* － Sun Solaris
  - tcp 用の *libinsck.sl* － HP 9000、RS/6000
  - tcp 用の *libinsck.so.a* － RS/6000

- *protocol* は、ネットワーク・プロトコルの名前です。有効な値は次のとおりです。
  - TCP/IP － “tcp”
  - TLI － “tli”

  この要素はドライバのプロトコルと一致させてください。
- *description* は、エントリの説明です。これはオプションです。

次の DRIVERS セクションでは、tli ドライバのエントリを示します。

```
[DRIVERS]
 libtli.so=tli The tli driver
```

注意 DRIVERS セクションでドライバを指定しない場合は、Open Client/Open Server 製品はプラットフォーム固有のデフォルト・ドライバを使用します。詳細については、表 B-2 を参照してください。

ディレクトリ・ドライバの追加

❖ *libtcl.cfg* にディレクトリ・ドライバを追加する

1 *provider* の値を選択します。任意の値を選択できます。

注意 エントリをデフォルト・ディレクトリ・ドライバにするには、そのエントリを DIRECTORY セクションの最初のエントリとして追加します。

2 *driver* の値を指定します。この値はプラットフォームによって異なります。
- IBM RS/6000 では、*libddce.so.a* を使用します。
- Sun Solaris と HP Tru64 UNIX では、*libddce.so* を使用します。
- HP/UX では、*libddce.sl* を使用します。

注意 HP/UX、IBM RS/6000、Sun Solaris では、オペレーティング・システム・プロバイダは OSF DCE 1.1 と互換性がなければなりません。

3 DIT ベースの値を指定します。この値は、DCE がサーバ・エントリの検索を開始するロケーションです。

4　DIT ベース・パスが DCE ディレクトリに存在することを確認します。

DCE 管理者はこの作業を行う必要がある場合があります。dcecp を実行し、directory list コマンドを使用して DIT ベース・ロケーションがあるかどうかを確認します。必要に応じて、dcecp ユーティリティの directory create コマンドを使用して DIT ベース・ロケーションを作成します。dcecp の詳細については、DCE のマニュアルを参照してください。

5　DIRECTORY セクションに移動し、次のフォーマットを使用してエントリを追加します。

```
provider=driver ditbase=Value
```

次に例を示します。

```
dce=libddce.so ditbase=/.:/subsys/sybase/testserver
```

異なる DIT ベースを使用する複数の DCE ドライバ・エントリを追加できます。複数のドライバ・エントリがあると、dscp_dce や dsedit_dce ツールを使用して DCE ディレクトリの異なるロケーションにあるエントリを表示したり、修正したりする場合に便利です。たとえば、次のようなエントリを追加する場合があります。

```
[DIRECTORY]
 dce=libddce.so ditbase=/.:/subsys/sybase/dataserver
 dcetest=libddce.so ditbase=/.:/subsys/sybase/testserver
```

セキュリティ・ドライバの追加

❖ libtcl.cfg にセキュリティ・ドライバを追加する

1　*provider* の値を指定します。この値は、オブジェクト識別子ファイル *$SYBASE/$SYBASE_OCS/config/objectid.dat* にリストされているセキュリティ・メカニズムのローカル名です。DCE セキュリティ・メカニズムのデフォルト・ローカル名は “dce” です。

2　*driver* の値を指定します。この値は、プラットフォームとセキュリティ・メカニズムによって異なります (表 B-2 (68 ページ) に、ドライバ名がリストされています)。

3　*init-string* の値を指定します。

DCE ドライバでは、*init-string* は次のフォームを使用します。

```
secbase=/.../cellname
```

ここでは、*cellname* には、DCE セルの名前が入ります。

CyberSafe Kerberos ドライバでは、*init-string* は次のフォームを使用します。

```
secbase=@realmname
```

realmname は、修飾されていない CyberSafe ユーザ名のデフォルトのレルム名です。

4 SECURITY セクションに移動し、次のフォーマットを使用してエントリを追加します。

```
provider=driver init-string
```

次に例を示します。

```
dce=libsdce.so secbase=/.:/sec/principal
```

ネットワーク・ドライバの追加

❖ libtcl.cfg にネットワーク・ドライバを追加する

1 *driver* の値を指定します。ドライバは次のとおりです。

- HP 9000 上 の TCP/IP の場合は、*libinsck.sl*
- RS/6000 上 の TCP/IP の場合は、*libinsck.so.a*
- Sun Solaris 上の TLI の場合は、*libtli.so*

2 *protocol* の値を指定します。この値は手順 1 で選択したドライバのプロトコルと一致していなければなりません。有効な値は次のとおりです。

- TCP/IP － “tcp”
- TLI － “tli”

3 *description* の値を選択します。任意の値を選択でき、省略も可能です。

4 DRIVERS セクションに移動し、次のフォーマットを使用してエントリを追加します。

```
driver=protocol description
```

次に例を示します。

```
libtli.so=tli The tli driver
```

注意 DRIVERS セクションでドライバを指定しない場合、Open Client/Server 製品はプラットフォーム固有のデフォルト・ドライバを使用します。詳細については、表 B-2 (68 ページ) を参照してください。

---

interfaces ファイル

interfaces ファイルには、サーバのネットワーク・ロケーションに関する情報が含まれています。

Open Client/Open Server は *interfaces* を限定機能のディレクトリ・サービスとして使用します。*interfaces* ファイルは、外部ディレクトリ・サービスに障害が発生した場合のデフォルトとしても機能します。

- Open Client は *interfaces* エントリの *query* 行に指定されているネットワーク情報を使用して、サーバに接続します。
- Open Server は *interfaces* エントリの *master* 行に指定されているネットワーク情報を使用して、クライアント接続要求を受信します。

interfaces ファイルは、インストール中に *$SYBASE*/interfaces として作成されます。Open Client/Server 製品は、*$SYBASE* 内で *interfaces* を探します。

アプリケーションは、デフォルトのロケーション以外で *interfaces* を探すことができます。詳細については、『Open Client Client-Library/C リファレンス・マニュアル』の「ct_config」と『Open Server Server-Library/C リファレンス・マニュアル』の「srv_props」を参照してください。

interfaces のエントリ

Open Client/Open Server 11.1 以降は *interfaces* エントリに標準フォーマットを使用します。

注意　Open Client/Open Server の上記以前のバージョンと SQL Server の現在のバージョンは、tli ベースのプラットフォーム用に 16 進数エントリ・フォーマットが必要です。この項では標準フォーマットを定義し、以前の “tli” フォーマットをいつ使用したらよいかを説明します。

標準フォーマット

interfaces エントリには、次のフォームを使用します。

```
# put comments here<newline>
 SERVERNAME[<tab>retry_count<tab>retry_delay]<newline>
 <tab>{master|query} protocol network host port<newline>
 <tab>[secmech mechanism1,..., mechanismn]<newline>
 <blank line>
```

各パラメータの意味は、次のとおりです。

- *SERVERNAME* は Open Client/Open Server が、どの *interfaces* エントリを読み込むのかを認識するときに使用するエイリアスです。*SERVERNAME* には、アルファベット (ASCII a-z、A-Z) で始まり、アルファベット、数字、アンダースコアだけで構成される 11 文字以内の名前を指定します。
- *retry_count* (オプション) には、クライアントが最初の接続に失敗したあと、サーバに接続しようとする回数を指定します。
- *retry_delay* (オプション) には、接続しようとする間隔を指定します。
- “master | query” には、次のように接続のタイプを指定します。
  - “master” は master 行を指定します。これはサーバ・アプリケーションがクライアント・クエリを受信するときに使用します。
  - “query” は query 行を指定します。これはクライアント・アプリケーションがサーバを探すときに使用します。

  interfaces エントリの master 行と query 行には、同じ情報が含まれています。dscp ユーティリティは各エントリに両タイプの行を作成します。結果のエントリはクライアントとサーバの両方が使用できます。
- *protocol* は、ネットワーク・プロトコルの名前です。有効な値は次のとおりです。
  - TCP/IP の場合は “tcp” － すべての UNIX プラットフォーム
  - IPX/SPX の場合は “spx” － UnixWare
  - DECnet の場合は “decnet” － HP Tru64 UNIX
- *network* は、ネットワークの記述子です。

  Open Client/Open Server は、現時点では *network* を使用していません。*network* はプレースホルダであり、Sybase は今後この情報を定義します。
- *host* は、サーバが稼働しているノードやマシンのネットワーク名です。*host* に指定できる最大文字数はエントリで指定されるプロトコルによって異なります。
  - TCP/IP での最大文字数は 32 文字です。
  - DECnet での最大文字数は 6 文字です。

  /bin/hostname コマンドを使用して、ログインするマシンのネットワーク名を調べます。

- *port* は、クエリを受け取るためにサーバが使用するポートです。TCP/IP と DECnet プロトコルはそれぞれ次のようにこの要素を指定します。
  - TCP/IP：有効なポート番号の範囲は 1025 から 65535 までです。ただし、1025 ～ 7009、9535、17007 は登録されたポート番号で、システムですでに使用されている可能性があります。
  - DECnet：有効なオブジェクト番号は 128 から 253 までの範囲です。オブジェクト名も有効です。

  netstat コマンドを使用して、どのポート番号が使用されているかを確認してください。
- オプションの SECMECH 行には、サーバがサポートするセキュリティ・メカニズムをリストするときに使用する識別子が含まれています。
- *mechanism1,..., mechanismn* はサーバがサポートするセキュリティ・メカニズムです。カンマをセパレータとして使用して複数のセキュリティ・メカニズムを指定できます。

  セキュリティ・メカニズムはオブジェクト識別子としてリストされます。オブジェクト識別子は、グローバル・オブジェクト識別子ファイル内のセキュリティ・メカニズムのローカル名にマップしたグローバルにユニークな数字列です。

  オブジェクト識別子の詳細については、「objectid.dat ファイル」(88 ページ) を参照してください。

トランスポート・レイヤ・インタフェース・フォーマット

Open Client/Open Server 11.1 では *interfaces* エントリに標準フォーマットを使用しますが、tli の 16 進数フォーマットを使用するエントリも認識します。

dscp では、次のように入力して、トランスポート・タイプとして tli フォーマットを指定できます。

```
tli protocol
```

詳細については、「サーバ・エントリの追加と変更」(42 ページ) を参照してください。

interfaces ファイルの編集

dscp、または vi などのオペレーティグ・システム・エディタを使用して *interfaces* を編集します。

dscp を使用して *interfaces* ファイルを編集すると、dscp が入力するアドレス文字列を正しくフォーマットするので、処理が簡単になります。dscp を使用して *interfaces* ファイルを編集する手順については、「第 7 章 dscp の使用」を参照してください。

スタンバイ・サーバ・アドレッシング

interfaces ファイルを設定すると、スタンバイ・サーバ・アドレッシングが可能になります。スタンバイ・サーバ・アドレッシングを使用すると、Open Client は、最初の接続に失敗した場合、代替サーバに接続できます。

たとえば、次に示す *interfaces* エントリは、“violet” というマシン上のポート番号 1025 のサーバにアプリケーションをダイレクトします。このサーバが使用できない場合、接続は失敗します。

```
#
 BETA
     query tcp hp-ether violet 1025
     master tcp hp-ether violet 1025
     secmech 1.3.6.1.4.1.897.4.6.1
```

ただし、BETA エントリに複数の *query* 行がある場合、Open Client は、最初の接続に失敗すると、リストされている次のサーバに自動的に接続しようとします。この *interfaces* エントリは、次のように表示されます。

```
#
 BETA
 query tcp hp-ether violet 1025
 query tcp hp-ether plum 1050
 query tcp hp-ether mauve 1060
 master tcp hp-ether violet 1025
 secmech 1.3.6.1.4.1.897.4.6.1
```

注意 *interfaces* エントリの *SERVERNAME* の要素はエイリアスであり、実際のサーバをユニークに識別しません。ホストとポートの要素は、サーバをユニークに識別します。

前述の例では、Open Client は、ポート 1025 の “violet” への接続に失敗するとポート 1050 の “plum” にというように、次の query 行にリストされているサーバに接続しようとします。

代替サーバの番号は、サーバの *interfaces* エントリにリストされることもありますが、各代替サーバは同一の *interfaces* ファイルにリストされていなければなりません。

ocs.cfg ファイル

ランタイム設定ファイル *ocs.cfg* は Client-Library アプリケーションが使用し、次のものを設定します。

- プロパティ値
- サーバ・オプション値
- サーバ機能
- デバッグ・オプション

ocs.cfg を使用することによって、アプリケーションで値を設定するルーチンを呼び出す必要がなくなり、コードを再コンパイルすることなくアプリケーションの設定を変更できます。

デフォルトでは、Client-Library は *ocs.cfg* を読み込みませんが、*$SYBASE/$SYBASE_OCS/config* にファイル名がある場合、*ctlib* ベースのすべてのアプリケーションはファイルを読み込もうとします。Client-Library がこのファイルを使用できるように、アプリケーションでプロパティを設定する必要があります。

ファイル構文と、ファイルに設定できるプロパティについては、『Open Client Client-Library 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」を参照してください。

付 録　C

ローカライゼーション

ローカライゼーションとは、特定の言語を使用して、その言語を使用する国の慣習に従って実行できるように、アプリケーションを初期化するプロセスです。

この章では、システム設定の観点からローカライゼーションとローカライゼーション・ファイルを説明します。ローカライゼーションに関するプログラミングの問題については、Open Client/Open Server の『開発者用国際化ガイド』を参照してください。

この付録の内容は、次のとおりです。

ローカライゼーション・プロセスの概要

Open Client/Open Server アプリケーションのローカライズには次の 2 つの方法があります。

- 初期ローカライゼーション値の使用
- 初期ローカライゼーション値とカスタム・ローカライゼーション値の使用

すべての Open Client/Open Server アプリケーションは初期ローカライゼーション値を使用します。これは、実行時に決定されます。

さらにアプリケーションの実行中、特定の時点でローカライズする必要があった場合、Open Client/Open Server アプリケーションはカスタム・ローカライゼーション値も使用できます。カスタム・ローカライゼーション値は、実行時に設定された初期ローカライゼーション値を上書きします。

ローカライゼーション時に使用する環境変数

Open Client と Open Server は環境変数を使用して、*locales.dat* ファイルでどのロケール名を探すかを決定します。Open Client と Open Server は必ず次の環境変数を検索します。

- LC_ALL
- LANG (LC_ALL が設定されていない場合)

カスタム・ローカライゼーション値を設定する場合、Open Client と Open Server は表 C-1 に示されている環境変数を検索することもあります。

表 C-1: ローカライゼーションで使用する環境変数

環境変数	説明	使用
LC_ALL	メッセージ、データ型変換、ソートに使用する言語、文字セット、照合順	初期ローカライゼーション、カスタム・ローカライゼーション
LANG	メッセージ、データ型変換、ソートに使用する言語、文字セット、照合順 Open Client/Open Server 製品は、LC_ALL 環境変数を見つけることができない場合には LANG 環境変数を検索する。	初期ローカライゼーション
LC_COLLATE	文字データのソートと比較を行うときに使用する照合順 (ソート順)	カスタム・ローカライゼーション
LC_CTYPE	データ型変換に使用する文字セット	カスタム・ローカライゼーション
LC_MESSAGE	メッセージに使用する言語。	カスタム・ローカライゼーション
LC_TIME	日付と時刻のフォーマット、ネイティブ言語での名前、月と日の省略形などの日時文字列に使用する日付と時刻のデータ表現	カスタム・ローカライゼーション

アプリケーションがカスタム・ローカライゼーション時にどの環境変数を使用するかについては、Open Client/Open Server の『開発者用国際化ガイド』を参照してください。

ローカライズされたアプリケーションを実行する前に、次の点に注意してください。

- *locales.dat* ファイルに、アプリケーションが使用するローカライゼーション値を反映したエントリが入っていることを確認してください。入っていない場合は、該当するエントリを追加してください。
- アプリケーションが使用するローカライゼーション・ファイルがインストールされていることを確認してください。
  - ローカライズしたメッセージ・ファイルは *$SYBASE/locales/message* ディレクトリにあります。

- 照合順ファイルは *$SYBASE/charsets* ディレクトリにあります。

すべての Open Client/Open Server 製品には、最低 1 つの言語と、1 つまたは複数の文字セットと照合順 (ソート順) をサポートするファイルが含まれています。インストール時に、これらのファイルは *$SYBASE* ディレクトリ構造の適切なロケーションにロードされます。Open Client または Open Server アプリケーションを設定するときには、上記のディレクトリに、ユーザ・サイトとユーザ・アプリケーションに適切なファイルが入っていることを確認してください。

ローカライゼーション・ファイル

実行時に、Open Client/Open Server アプリケーションは外部ファイルからローカライゼーション情報をロードします。*$SYBASE* ディレクトリの 3 つのディレクトリには、これらのファイルが入っています。

- *locales* ディレクトリは次のディレクトリとファイルから構成されます。
  - 言語、文字セット、照合順にロケール名をマップする *locales.dat* ファイル
  - すべての製品用のローカライズされたエラー・メッセージが入っている *message* サブディレクトリ。このディレクトリは言語名別に編成されています。
  - 以前のリリースの Open Client/Open Server ソフトウェアとの互換性のために用意されている *language_name* サブディレクトリ。このディレクトリには、ローカライズされたメッセージ・ファイルが文字セット別に編成されて入っています。
- *charsets* ディレクトリには、サポートされている各文字セットのサブディレクトリが入っています。それぞれのサブディレクトリには、文字セットのソート・ファイルと変換ファイルが入っています。
- *config* ディレクトリには、次のファイルが入っています。
  - 文字セットや言語などのオブジェクトのグローバル名をプラットフォームに依存したローカルな名前にマップする *objectid.dat* ファイル
  - 必要に応じて、ソース Unicode と置換するためのニーモニック文字列が入っている *mnemonic.dat* ファイル

locales ディレクトリ

locales ディレクトリには、アプリケーションがローカライゼーション情報をロードするときに使用するファイルが入っています。また、言語固有のメッセージ・ファイルも入っています。

locales.dat ファイル

ロケール・ファイル (*locales.dat*) は、プラットフォームに依存するロケール情報を Sybase 独自のフォーマットで提供します。このファイルは、言語、文字セット、照合順とロケール名を対応させます。

使用方法

Open Client/Open Server アプリケーションは *locales.dat* を使用して、どのローカライゼーション情報をロードするかを決定します。*locales.dat* ファイルは Open Client/Open Server アプリケーションのためのローカライゼーション情報を格納していますが、ローカライズされた実際のメッセージまたは文字セットの情報は入っていません。

***locales.dat* のロケーション**

locales.dat ファイルは *$SYBASE/locales* ディレクトリにあります。*$SYBASE/locales* ディレクトリ構造図については、「ローカライゼーション・ファイル」(81 ページ) を参照してください。

***locales.dat* のセクションとエントリ**

locales.dat は、プラットフォーム固有のセクションで構成され、各セクションには事前に定義されたロケール定義エントリが入っています。これらのエントリはプラットフォームによって異なりますが、すべてのセクションには「デフォルト」ロケールを定義するエントリが指定されています。

ロケール定義エントリの形式は、次のとおりです。

```
locale = locale_name, language_name, charset_name
[,sortorder_name]
```

各パラメータの意味は、次のとおりです。

- *locale_name* はロケール定義の名前です。*locale_name* のデフォルト値は、ベンダ指定であり、POSIX 用語規定に基づいています。*locales.dat* ファイルの末尾にあるコメントには、ロケール名の POSIX 値がリストされています。
- "," (カンマ) はファイルのリスト・セパレータ文字です。
- *language_name* は Sybase 製品が言語を認識するときに使用するサブディレクトリ名です。
- *charset_name* は Sybase 製品が文字セットを認識するときに使用するサブディレクトリ名です。
- *sortorder_name* は Sybase 製品が照合順を認識するときに使用するファイル名です (オプション)。

次の *locales.dat* ファイル・エントリでは、フランス語のロケールを指定しています。このロケールではソート順が指定されていないので、デフォルトのソート順である「バイナリ」が使用されます。

```
locale = fr.FR.88591, french, iso_1
```

locales.dat ファイルの例

locales.dat の次の部分は、プラットフォーム固有のセクションを示しています。

```
[aix]

      locale = C, us_english, iso_1
      locale = En_US, us_english, iso_1
      locale = en_US, us_english, iso_1
      locale = default, us_english, iso_1
      locale = japanese.sjis, japanese, sjis
      locale = japanese, japanese, eucjis
      locale = us_english.utf8, us_english, utf8
```

locales.dat の編集

locales.dat の事前に定義されたエントリがユーザのニーズに合わない場合は、vi などのオペレーティング・システムのテキスト・エディタを使用してファイルを編集します。

警告！ 編集を行う前に、元の *locales.dat* のコピーを作成してください。コピーを作成しておくと、編集したファイルで問題が発生した場合に役立ちます。また、プラットフォームのエントリを調べて、エントリがすでにあるかどうかも確認してください。

---

locales.dat を編集して次のことを行います。

- 「デフォルト」ロケール定義を変更します。
- ロケール定義を追加します。
- Sybase 以外のソフトウェアが使用するロケール名に合わせます。たとえば、次のように Sybase であらかじめ定義されているロケール名は “fr” です。

  ```
  locale = fr, french, iso_1
  ```

  Sybase 以外のアプリケーションでは LC_ALL 環境変数に “french” の値が必要な場合は、ロケール名を次のように変更します。

  ```
  locale = french, french, iso_1
  ```

locales.dat ファイルに新しいエントリを追加したり、既存のエントリを変更するには、次の手順に従ってください。

1 *locale_name* に使用する値を選択します。

2 *language_name* の値を決定します。

Sybase 言語モジュールがインストールされると、Sybase ディレクトリ・ツリーの *locales*/*message* ディレクトリに言語のサブディレクトリが作成されます。*language_name* はこのサブディレクトリの名前と一致している必要があります。

3 *charset_name* の値を決定します。

Sybase の言語モジュールがインストールされると、Sybase ディレクトリ・ツリーの *charsets* ディレクトリに、サポートされているそれぞれの文字セット用のサブディレクトリが作成されます。*charset_name* は、これらのサブディレクトリ名のうちの 1 つと一致している必要があります。

4 バイナリ以外のソート順を選択する場合は、*sortorder_name* の値を決定します。

charsets/*charset_name* サブディレクトリには、文字セットのソート順 (**.srt*) ファイルがあります。*sortorder_name* は、これらのファイル名のうちの 1 つと一致している必要があります (*.srt* サフィックスの部分は指定しません)。

5 *locales.dat* ファイルの該当するプラットフォーム固有セクションで、該当するエントリを入力または変更します。

ローカライゼーション環境変数 (LC_ALL、LC_CTYPE、LC_MESSAGE、LC_TIME、LANG) を適切に更新します。

新しいロケール名をすでに追加していて、既存のアプリケーションが cs_locale 呼び出しでこの新しい名前を使用するようにしたい場合は、アプリケーションを適切に編集して再コンパイルします。

注意 アプリケーションがエントリを使用しなくなっても、*locales.dat* からそのエントリを削除する必要はありません。エントリを削除する場合は、そのエントリを使用するアプリケーションが 1 つもないことを確認してください。

ローカライズしたメッセージ・ファイル

警告！ ローカライズしたメッセージ・ファイルは編集しないでください。

ローカライズされたメッセージ・ファイルには、特定の言語で記述した製品メッセージが入っています。これらのメッセージ・ファイル (*locales/message/language_name* ディレクトリ内の **.loc* ファイル) を使用して、Open Client/Open Server アプリケーションはさまざまな言語でメッセージを生成できます。

すべての Open Client/Open Server 製品には、英語 (us_english) のメッセージ・ファイルが入っています。他の言語をサポートするためのファイルが入っている場合もあります。

新しい言語モジュールを購入してインストールする場合、インストール処理で *language_name* サブディレクトリが新規に作成され、新しい言語のメッセージ・ファイルが格納されます。

メッセージ・ファイル名はプラットフォームによって異なることもありますが、たいていは次のような名前になります。

- *cslib.loc* － CS-Library メッセージ
- *ctlib.loc* － Client-Library メッセージ
- *oslib.loc* － Server-Library メッセージ
- *blklib.loc* － Bulk Library メッセージ
- *bcp.loc* － Bulk Copy メッセージ
- *esql.loc* － Embedded SQL メッセージ

すべての Open Client/Open Server メッセージ・ファイルは Unicode ISO 10646 UTF-8 文字セットを使用します。

Open Client/Open Server 製品は、必要に応じてメッセージを UTF-8 から他の文字セットに変換します。

charsets ディレクトリ

charsets ディレクトリには、サポートする各文字セットの変換ファイルと照合順ファイルが入っています。

変換設定ファイル

各文字セットの変換設定ファイルには、変換処理に関する情報が入っています。

変換設定ファイルの使い方

クライアントとサーバが異なる文字セットを使用する場合は、文字セット間での変換が必要となります。Open Client/Open Server 製品には、各文字セットの変換をサポートするファイルが含まれています。

文字セットの変換設定ファイルには、変換に使用するモードとマップ不可能な文字に使用する置換文字が指定されています。

表 C-2 には、変換モードが説明されています。

表 C-2: コード化文字セット変換のモード

モード	説明
MATCH 出荷時に提供されているファイルにはこの値が入っている。	変換処理では、変換元と変換先とで一致している値が変換される。 変換元の文字のコードが正しくないか、またはマップ不可能である場合は、変換処理では、変換先の文字セットの変換設定ファイルに定義されている送信先置換文字を使用する。
BESTGUESS	変換処理では、変換元と変換先とで一致している値と近似の値が変換される。 変換元の文字のコードが正しくないか、またはマップ不可能である場合は、変換処理では、変換先の文字セットの変換設定ファイルに定義されている送信先置換文字を使用する。
MNEMONIC	変換元と変換先とで一致している値が変換される。変換元の値に対して一致している値がない場合、変換処理では変換先の値として Unicode のニーモニック文字列が使用される。適切なニーモニック文字列がない場合は、変換処理では変換先の値として Unicode の 16 進数文字列が使用される。 変換元の文字のコードが正しくない場合は、変換処理では、変換先の文字セットの変換設定ファイルに定義されている送信先置換文字を使用する。

文字セット変換処理の詳細については、Open Client/Open Server の『開発者用国際化ガイド』を参照してください。

変換設定ファイルのロケーション

各文字セットに、変換設定ファイルがあります。このファイルは *$SYBASE/charsets/charset_name/charset_name.cfg* にあります。

$SYBASE/charsets ディレクトリ構造図については、「ローカライゼーション・ファイル」(81 ページ) を参照してください。

変換設定ファイル・エントリ

変換セクションには、特定の文字セットへの変換がどのように行われるかを記述するエントリが入っています。変換セクションのエントリは、テーブル駆動型の変換かアルゴリズム駆動型の変換かを示す場合があります。

テーブル駆動型の変換の場合、エントリの形式は次のようになります。

```
[conversion]
     convertto = dest_charset, table, mode, replacement_char
```

各パラメータの意味は、次のとおりです。

- *dest_charset* は変換先の文字セットの名前です。
- "," (カンマ) はファイルのリスト・セパレータ文字です。
- table は変換がテーブル駆動型であることを示すキーワードです。
- *mode* は使用する変換モードです。テーブル駆動型の変換だけに適用されます。有効な値は次のとおりです。
  - MATCH
  - BESTGUESS
  - MNEMONIC

  各モードの詳細については、表 C-2 を参照してください。
- *replacement_char* は MATCH と BESTGUESS モード変換時に使用する送信先置換文字の 16 進 (0x プレフィクスなしの) コードです。

アルゴリズム駆動型の変換の場合、エントリの形式は次のようになります。

```
[conversion]
    convertto = dest_charset, sys_algorithm, multiplier
```

各パラメータの意味は、次のとおりです。

- *dest_charset* は変換先の文字セットの名前です。
- , (カンマ) はファイルのリスト・セパレータ文字です。
- sys_algorithm は変換処理が標準 Open Client/Open Server 変換アルゴリズムを使用することを示すキーワードです。
- *multiplier* は変換に使用する変換乗数を表す整数値です。この値は、変換時の文字列長の増加量の最大値を示しています。

変換設定ファイルの例

次に、変換設定ファイルの例を示します。

```
; Conversion Configuration File for iso_1 charset.
 [conversion]
 convertto = utf8, table, MATCH, 3F
 convertto = cp850, sys-algorithm, 1
 convertto = cp437, sys-algorithm, 1
 convertto = roman8, sys-algorithm, 1
 convertto = mac, sys-algorithm, 1
```

照合順ファイル

警告！ 照合順ファイルは編集しないでください。

システムが文字をソートする順序は、「照合順」または「ソート順」と呼ばれます。

Open Client/Open Server 製品には、さまざまな照合順をサポートするファイルが用意されています。これらのファイルはプラットフォームによって異なることがありますが、一般に次のようなファイルがあります。

- *binary.srt*
- *dictionary.srt*
- *noaccents.srt*
- *nocase.srt*
- *nocasepref.srt*

照合順は、*locales.dat* ファイル・エントリに指定されています。*locales.dat* ファイル・エントリに照合順が指定されていない場合は、バイナリ・ソート順を使用します。

照合順の詳細については、Open Client/Open Server の『開発者用国際化ガイド』を参照してください。

config ディレクトリ

config ディレクトリには、グローバル・オブジェクト識別子ファイル (*objectid.dat*) とニーモニック・ファイル (*mnemonic.dat*) が入っています。

objectid.dat ファイル

$SYBASE/config ディレクトリにある *objectid.dat* ファイルは、オブジェクトのローカル名をユニークなグローバル・オブジェクト識別子に対応させます。

オブジェクト識別子は、ドットで区切った一連の正の整数値です。この識別子は国際標準団体である CCITT と ISO が定義したネーミング・ツリーに基づいています。

objectid.dat のセクションとエントリ

objectid.dat ファイルはオブジェクト・クラスごとに 1 つのセクションで構成されています。

オブジェクト・クラス・エントリのフォームは次のとおりです。

```
[Object Class]
    object_identifier local_name1, ..., local_namen
```

各パラメータの意味は、次のとおりです。

- *Object Class* はセクション識別子です。
- *object_identifier* はグローバルにユニークなオブジェクト識別子です。
- *local_name1,..., local_namen* はカンマで区切ったオブジェクト識別子に対応するローカル名です。

objectid.dat の例

次の例は *objectid.dat* のセクションを示しています。

```
[charset]
     1.3.6.1.4.1.897.4.9.1.1 = iso_1
     1.3.6.1.4.1.897.4.9.1.2 = cp850
     1.3.6.1.4.1.897.4.9.1.3 = cp437
     1.3.6.1.4.1.897.4.9.1.4 = roman8
     1.3.6.1.4.1.897.4.9.1.5 = mac

[collate]
     1.3.6.1.4.1.897.4.9.3.50 = binary
     1.3.6.1.4.1.897.4.9.3.51 = dictionary
     1.3.6.1.4.1.897.4.9.3.52 = nocase
     1.3.6.1.4.1.897.4.9.3.53 = nocasepref
     1.3.6.1.4.1.897.4.9.3.54 = noaccents

[secmech]
     1.3.6.1.4.1.897.4.6.1 = dce
     1.3.6.1.4.1.897.4.6.2 = nds
     1.3.6.1.4.1.897.4.6.3 = NTLM
     1.3.6.1.4.1.897.4.6.6 = csfkrb5
```

objectid.dat の編集

オブジェクトのローカル名を変更する場合は、*objectid.dat* を vi などのオペレーティング・システム・エディタを使用して編集します。

mnemonic.dat ファイル

mnemonic.dat ファイルには、文字セット変換中に必要に応じて、マップ不可能な変換元の文字を置き換えるために使用できる POSIX ニーモニック文字列が入っています。

mnimonic.dat には、UCS-2 <-> ニーモニック文字列変換だけが入っています。出荷時の *mnemonic.dat* ファイル内にある各 Unicode ニーモニックは、Unicode 文字を表す XPG4 文字の文字列です。*mnemonic.dat* ファイルは、POSIX ニーモニック文字列を Unicode UCS-2 文字コードと対応させます。このニーモニック文字列は XPG4 Portable Character Set の文字を使用しているので、どの変換先の文字セットに使用するにも適しています。

mnemonic.dat のロケーション

mnemonic.dat は *$SYBASE/config* ディレクトリにあります。*$SYBASE/charsets* ディレクトリ構造図については、「ローカライゼーション・ファイル」(81 ページ) を参照してください。

mnemonic.dat の使い方

変換先の文字セットの変換設定ファイル (*charset.cfg*) で “Mnemonic” モードが指定されているときだけ、*mnemonic.dat* が使用されます。この場合、変換時に *mnemonic.dat* が次のように使用されます。

1 変換元の文字が変換先の文字セットにはマップ不可能であるとわかった場合、Sybase ソフトウェアは変換元の文字を Unicode UCS-2 に変換します。

2 Sybase は *mnemonic.dat* 内で UCS-2 コード化を検索して、変換先のデータ・ストリームでその UCS-2 コードと対応するニーモニック文字列を使用します。

3 *mnemonic.dat* ファイルに適切な文字列が含まれていない場合は、送信先データ・ストリームには Unicode UCS-2 16 進文字列を使用します。

文字セット変換処理の詳細については、Open Client/Open Server の『開発者用国際化ガイド』を参照してください。

mnemonic.dat のエントリ

mnemonic.dat には、UCS-2 コードをニーモニック文字列と対応させるエントリが入っています。

ニーモニック・セクションのエントリの形式は次のようになります。

```
mnem = <mnem_string> <UCS-2_encoding> comment
```

各パラメータの意味は、次のとおりです。

- < は無視されます。
- *mnem_string* はニーモニック文字列を表す XPG4 の文字列です。
- > はファイルのリスト・セパレータ文字です。
- *UCS-2_encoding* は文字の UCS-2 コードです。
- *comment* はコメント文字列です。

ニーモニック・セクションのエントリは、その他の Sybase ローカライゼーション・ファイルのエントリとは多少異なります。これは、*mnemonic.dat* ファイルでは標準 POSIX 定義を使用しているためです。

mnemonic.dat の例

次に、ニーモニック・ファイルの例を示します。

```
[file format]
      version = 11.0
      escape = /
      list_separator = >

[copyright]

     copyright = "Copyright ... ."

[mnemonics]
      mnem = <NU> <U0000> NULL (NUL)
      mnem = <SH> <U0001> START OF HEADINGS(SOH)
      mnem = <SX> <U0002> START OF TEXT (STX)
      mnem = <EX> <U0003> END OF TEXT (ETX)
      mnem = <ET> <U0004> END OF TRANSMISSION (EOT)
      mnem = <EQ> <U0005> ENQUIRY (ENQ)
      mnem = <AK> <U0006> ACKNOWLEDGE (ACK)
      mnem = <BL> <U0007> BELL (BEL)
      mnem = <BS> <U0008> BACKSPACE (BS)
      .
      .
      .
```

mnemonic.dat への文字列の追加

Sybase が出荷する *mnemonic.dat* には、すべての文字セットのすべての文字のニーモニック文字列が含まれているわけではありません。そのため、必要な文字列が *mnemonic.dat* に含まれていない場合は、vi などのオペレーティング・システムのエディタを使用して文字列を追加できます。

ftp サイトの *unicode.org* には、既存の文字列の更新だけでなく、新しい Unicode ニーモニック文字列についての情報もあります。

付 録　D

DCE セキュリティ・サービス

この章では、DCE セキュリティ・ドライバによってサポートされるセキュリティ・サービスをリストし、DCE セキュリティ・ドライバを使用するときに必要な設定作業について説明します。この章の内容は、次のとおりです。

Open Client と Open Server のセキュリティ・サービス・アーキテクチャの概要については、「第 6 章 セキュリティ・サービスの使い方」を参照してください。

サポートされているセキュリティ・サービス

DCE セキュリティ・メカニズムでは、次のセキュリティ・サービスを提供しています。

- ネットワーク認証
- 相互認証
- データの整合性
- データの機密保持
- リプレイの検出
- 順序不整合の検出
- クレデンシャルの委任
- データ作成
- チャネル・バインド

これらのセキュリティ・サービスの詳細については、『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。

Open Client と Open Server のための DCE の設定

ここでは、DCE セル内で実行する Open Client/Open Server アプリケーションで DCE セキュリティを使用するのに必要な管理作業について説明します。

ここで説明する DCE 管理作業の中には、DCE dcecp ツールを使用しなければならないものもあります。このツールについては、使用している DCE のマニュアルを参照してください。

Open Server アプリケーションと DCE セキュリティ

カスタム Open Server アプリケーションまたは Security Guardian サーバは DCE セルで実行できます。

サーバとそのクライアントがネットワークを介して通信するには、「第 3 章 Open Server の基本設定」で説明した通常の設定作業を行ってください。サーバとそのクライアントが DCE セキュリティ・サービスを使用するには、次の追加の設定作業が必要です。

1 サーバ をどの DCE プリンシパルとして実行するかを決定します。

サーバは DCE ルート・ユーザ */.:/hosts/hostname/self* として実行できます。*hostname* はサーバが稼働しているコンピュータです。サーバに新しいプリンシパルを作成することもできます。

必要に応じて、DCE dcecp ツールの user create コマンドを使用して、新しいプリンシパルを作成します。この場合、コマンド・オプションに、新しいプリンシパルがサーバとして動作できるように指定する必要があります。

2 サーバをルート・プリンシパルとして実行しない場合は、サーバ・プリンシパルの DCE keytab ファイルを作成する必要があります。

DCE keytab ファイルは、プリンシパルのパスワードが暗号形式で入っているオペレーティング・システム・ファイルです。DCE の dcecp ユーティリティの keytab create コマンドを使用して keytab ファイルを作成します。Open Server を起動するオペレーティング・システム・ユーザは、keytab ファイルに対する読み込みパーミッションを持っていなければなりません。運用環境では、このファイルへのアクセスを制御する必要があります。keytab ファイルの読み込みを許可されているユーザは、使用しているサーバになりかわるサーバを作成できます。

3 DCE セキュリティ・ドライバが *libtcl.cfg* の [SECURITY] セクションに設定されているかどうかを確認してください。詳細については、「SECURITY セクション」(68 ページ) を参照してください。

4　サーバ・プリンシパル名がサーバのネットワーク名と同じではない場合には、サーバを起動するときに、プリンシパル名を指定します。

Open Server のネットワーク名は *interfaces* または DCE ディレクトリ・サービスでの名前です。プリンシパル名がネットワーク名と一致しない場合は、プリンシパル名を別に指定する必要があります。

カスタム Open Server アプリケーションでは、SRV_S_SEC_PRINCIPAL Server-Library プロパティを設定してプロパティ名を指定します。

Security Guardian ユーザは、-R コマンドライン・オプションでサーバのプリンシパル名を指定できます。

5　サーバが DCE ルート・ユーザ (*/.:/hosts/hostname/self*) として実行されない場合は、サーバの起動時に、DCE keytab ファイル (上記の手順 2 を参照) のロケーションを指定します。

カスタム Open Server アプリケーションでは、SRV_S_SEC_KEYTAB Server-Library プロパティを設定して keytab ファイルのロケーションを指定します。

Security Guardian ユーザは、-K コマンドライン・オプションでサーバのプリンシパル名を指定できます。

Client-Library アプリケーションと DCE セキュリティ

クライアント・アプリケーションがセキュリティ・サービスを使用する方法の概要については、「Client-Library と セキュリティ・サービス」(36 ページ) を参照してください。

DCE セキュリティ・サービスを使用するクライアント・アプリケーションでは、次の点に注意してください。

- クライアント・アプリケーションには、サーバ・プリンシパル名がサーバのネットワーク名と同じでない場合、プリンシパル名を指定する必要があります。

  DCE セキュリティを使用する場合、DCE は常にサーバのプリンシパル名を認証します。正しいサーバ・プリンシパル名を入力しないと、接続はオープンできません。デフォルトでは、Client-Library はプリンシパル名をネットワーク名と同じであると見なします。

  Client-Library アプリケーションには、CS_SEC_SERVERPRINCIPAL 接続プロパティを設定してサーバ・プリンシパル名を指定します。isql および他の Sybase クライアント・ユーティリティでは、-R コマンド・ライン・オプションでサーバ・プリンシパル名を指定できます。

- クライアント・アプリケーションは、すでに作成されているデフォルトの DCE クレデンシャルを使用するか、DCE keytab ファイルを使用して新しいクレデンシャルを入手して、サーバに接続する必要があります。

  DCE ユーザは、DCE dce_login ツールでデフォルトの DCE クレデンシャルを入手します。Client-Library アプリケーションは CS_USERNAME 接続プロパティを設定しないことによってデフォルトのクレデンシャルを使用します。isql などの Sybase クライアント・ユーティリティでは、-U コマンド・ライン・オプションを省略して、デフォルトのクレデンシャルを使用して接続します。

  新しいクレデンシャルを得るには、接続に使用する DCE ユーザの暗号化されたパスワードを含む有効な DCE keytab ファイルに対する読み込みアクセス権が必要です。keytab create コマンドを使用して、DCE dcecp ユーティリティで keytab ファイルを作成できます。

  Client-Library アプリケーションは、CS_USERNAME プロパティを DCE ユーザ名に設定し、CS_SEC_KEYTAB プロパティを対応する DCE keytab ファイルの名前に設定することによって、新しいクレデンシャルを得ることができます。

  isql および他の Sybase クライアント・ユーティリティでは、-U コマンド・ライン・オプションを使用してユーザ名を指定でき、-K コマンド・ライン・オプションを使用して keytab ファイル名を指定できます。

注意 ユーザの認証に DCE keytab ファイルの名前も入力する場合以外は、ユーザ名を指定しないでください。

付 録 E

CyberSafe Kerberos セキュリティ・サービス

この章では、CyberSafe Kerberos セキュリティ・ドライバによってサポートされるセキュリティ・サービスをリストし、CyberSafe Kerberos セキュリティ・ドライバを使用するのに必要なシステム設定作業を説明します。

注意 DB-Lib は Kerberos をサポートしていません。

この付録の内容は、次のとおりです。

Open Client と Open Server のセキュリティ・サービス・アーキテクチャの概要については、「第 6 章 セキュリティ・サービスの使い方」を参照してください。

サポートされているセキュリティ・サービス

CyberSafe Kerberos セキュリティ・メカニズムは、次のサービスを提供します。

- ネットワーク認証
- 相互認証
- データの整合性
- データの機密保持
- リプレイの検出
- 順序不整合の検出

これらのセキュリティ・サービスの詳細については、『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。

CyberSafe Kerberos の設定

- 次のように、CyberSAFE ソフトウェアをシステムにインストールします。Open Client/Open Server 12.0 の場合は CyberSafe Challenger 5.3.1 以降、Open Client/Open Server 12.5 以降の場合は CyberSafe TrustBroker をインストールします。
- CyberSafe Application Development Kit バージョン 1.1。
- ct_con_props を使用してクレデンシャル (希望のセキュリティ機能) を設定したり、クレデンシャル・プロパティを設定しないでデフォルト・クレデンシャルを使用します。
- *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。
- アプリケーションは、サーバに接続するのに、すでに作成されているユーザ・クレデンシャルを使用しなければなりません。つまり、アプリケーションのユーザはクライアント・アプリケーションを実行する前に、CyberSafe にログインする必要があります。
- CyberSafe ユーティリティ kinit を使用して CyberSafe セキュリティ・メカニズムにログインし、Client-Library アプリケーションを実行します。
- ユーザ名を入力する場合、その名前はそのユーザ・クレデンシャルと一致していなければなりません。ユーザ名を入力しないと、Client-Library はそのユーザの CyberSafe クレデンシャルに対応するユーザ名を使用してサーバに接続します。
- これらの環境変数では、クレデンシャル・キャッシュ・ファイル、設定ファイル、レルム・ファイルへのパスを設定します。対応するファイルがデフォルト以外のディレクトリにある場合は、環境変数をファイルのフル・パスに設定します。
  - *CSFC5CCNAME* − クレデンシャル・キャッシュ・ファイル
  - *CSFC5CONFIG* − 設定ファイル
  - *CSFC5REALMS* − レルム・ファイル

  詳細については、CyberSafe のマニュアルを参照してください。
- Client-Library アプリケーションの実行中は、*gssapi.dll* と *csfc516.dll* の 2 つのファイルがパスに含まれていなければなりません。これら 2 つの DLL は Sybase によっては提供されていませんが、CyberSafe 製品には含まれています。これら 2 つの DLL が CyberSafe 製品に含まれていない場合は、CyberSafe に連絡して GSS-API ライブラリを入手します。
- CyberSafe Kerberos セキュリティ・サービスを使用する Client-Library アプリケーションをコンパイルするときに、余分なフラグは必要ありません。

- Open Client/Open Server と CyberSafe を設定したら、isql を使用して設定を検査できます。isql の [ファイル] - [セキュリティ] がグレー表示されている場合は、セキュリティ・サービスは利用できません。Open Client/Open Server と CyberSafe が正しく設定されているかどうかを確認してください。

サンプル・プログラムの設定例と実行例については、*SYBASE_OCS/sample/srvlibrary* ディレクトリの *README.SEC* を参照してください。

Open Server アプリケーションと CyberSafe Kerberos

CyberSafe Kerberos セキュリティを使用して、カスタム Open Server アプリケーションまたは Security Guardian サーバを実行できます。サーバとそのクライアントがネットワークを介して通信するには、「第 3 章 Open Server の基本設定」で説明している通常の設定作業を行ってください。サーバとそのクライアントが CyberSafe Kerberos セキュリティ・サービスを使用する場合は、次の追加の設定作業を行ってください。

1　サーバをどの CyberSafe Kerberos プリンシパルとして実行するかを決定します。

add コマンドを使用して、CyberSafe kadmin ユーティリティで新しいプリンシパルを作成できます。プリンシパルはサーバとして動作するようにしてください。

2　サーバ・プリンシパルが CyberSafe Kerberos サーバ・キー・テーブル・ファイルにキーを持っていない場合は、ext コマンドを使用して CyberSafe kadmin ユーティリティでキーを 1 つ作成します。サーバを起動するオペレーティング・システム・ユーザがサーバ・キー・テーブル・ファイルでの読み込みパーミッションを持っていることを確認します。運用環境では、キー・テーブル・ファイルへのアクセスを制御してください。このファイルを読み込みできるユーザは、使用しているサーバになり代わるサーバを作成できます。

3　CyberSafe Kerberos セキュリティ・ドライバが *libtcl.cfg* の [SECURITY] セクションに設定されていることを確認します。詳細については、「SECURITY セクション」(68 ページ) を参照してください。

4　CSFC5KTNAME　環境変数をサーバ・プリンシパル用のキーがあるキー・テーブル・ファイルの名前に設定します (項目　2 を参照)。サーバ・キー・テーブル・ファイルが CyberSafe システムのデフォルト以外のロケーションにある場合は、CyberSafe ランタイム・ライブラリでは、この環境変数が設定されている必要があります。

5 共有ライブラリ・ファイル (Sun Solaris 2.x では *libgss.so*、IBM RS/6000 では *libgss.so.a*、HP-UX では *libgss.sl*) を共有ライブラリ・パス (Sun Solaris 2.x では LD_LIBRARY_PATH、IBM RS/6000 では LIBPATH、HP-UX では SHLIB_PATH) で指定されているディレクトリに置かなければなりません。

これによって、クライアントは実行時にこの共有ライブラリ・ファイルを見つけることができます。この共有ライブラリ・ファイルは、CyberSafe インストールの *lib* サブディレクトリにも配置できます。ただしこれは、このサブディレクトリが共有ライブラリ・パスにある場合に限ります。

この共有ライブラリは Sybase によって提供されるのではなく、特定の CyberSafe 製品に含まれています。この共有ライブラリが CyberSafe 製品に含まれていない場合は、CyberSafe に連絡して GSS-API ライブラリを入手してください。

6 サーバを起動したら、プリンシパル名がネットワーク名と一致しない場合は、ネットワーク名に加えてプリンシパル名を指定します。DSLISTEN 環境変数をネットワーク名に設定した場合は、ネットワーク名を指定する必要はありません。

Open Server のネットワーク名は *interfaces* または DCE ディレクトリ・サービスでの名前です。

カスタム Open Server アプリケーションでは、SRV_S_SEC_PRINCIPAL Server-Library プロパティを設定してプロパティ名を指定します。

CyberSafe セキュリティを使用しているときは、-K オプションを使用してキー・テーブルを指定することはできません。CSFC5KTNAME 環境変数を使用しなければなりません (項目 4 を参照)。

Client-Library アプリケーションと CyberSafe Kerberos

クライアント・アプリケーションがセキュリティ・サービスを使用する方法の概要については、「Client-Library と セキュリティ・サービス」(36 ページ) を参照してください。CyberSafe Kerberos セキュリティ・サービスを使用するクライアント・アプリケーションでは、次の点に注意してください。

- アプリケーションは、サーバに接続するのに、すでに作成されているユーザ・クレデンシャルを使用しなければなりません。つまり、アプリケーションのユーザはクライアント・アプリケーションを実行する前に、CyberSafe にログインする必要があります。UNIX では、CyberSafe kinit ユーティリティを使用して、CyberSafe にログインしてください。
- ユーザ名を入力する場合、その名前はそのユーザ・クレデンシャルと一致していなければなりません。ユーザ名を入力しないと、Client-Library はそのユーザの CyberSafe クレデンシャルに対応するユーザ名を使用してサーバに接続します。

付 録　F

Open Client/Open Server の SSL (Secure Socket Layer)

この章では、Open Client/Open Server の SSL サポートと、SSL プロトコルの使用に必要なシステム設定作業について説明します。この章の内容は、次のとおりです。

Open Client と Open Server のセキュリティ・サービス・アーキテクチャの概要については、「第 6 章 セキュリティ・サービスの使い方」を参照してください。

SSL の概要

SSL は、クライアントからサーバへ、およびサーバからサーバへワイヤ・レベルまたはソケット・レベルで暗号化されたデータを送信するための、業界標準です。サーバとクライアントは一連の I/O ラウンド・トリップを交換し、安全な暗号化セッションをネゴシエートして合意してから、SSL 接続が確立されます。これは、「SSL ハンドシェイク」と呼ばれています。次の項で説明します。

SSL ハンドシェイク

クライアント・アプリケーションが接続を要求すると、SSL 対応サーバが証明書を提示し、ID を証明してから、データを送信します。基本的に、SSL ハンドシェイクは次の手順によって構成されています。

- クライアントはサーバに接続要求を送信します。要求には、クライアントがサポートしている SSL (または TLS: Transport Layer Security) オプションが含まれています。

- サーバは、証明書とサポートされている CipherSuites を返します。これには、SSL/TLS サポート・オプション、キー交換で使用されるアルゴリズム、デジタル署名が含まれています。
- クライアントとサーバがお互いに CipherSuite に合意すると、安全で暗号化されたセッションが確立されます。

SSL ハンドシェイクと SSL/TLS プロトコルの詳細については、Internet Engineering Task Force Web サイト (http://www.ietf.org) を参照してください。

Open Client/Open Server がサポートしている CipherSuites の詳細については、『Open Client Client-Library リファレンス・マニュアル』を参照してください。

Open Client/Open Server の SSL セキュリティ・レベル

SSL には、いくつかのセキュリティ・レベルがあります。

- SSL 対応サーバへの接続を確立すると、サーバは接続対象のサーバであることを自己認証し、暗号化された SSL セッションが開始されてからデータが送信されます。
- SSL セッションが確立されると、ユーザ名とパスワードが暗号化された安全で接続によって送信されます。
- サーバ証明書のデジタル署名を比較して、サーバから受信したデータが転送中に変更されたかどうかを判断します。

SSL フィルタ

SSL 対応 Adaptive Server への接続を確立すると、SSL セキュリティ・メカニズムが *interfaces* ファイルの master 行と query 行にフィルタとして設定されます。TCP/IP 接続の上層に位置する Open Client/Open Server プロトコル層として SSL は使用されます。

SSL フィルタは、*interfaces* ファイル (Windows では *sql.ini*) の、SECMECH (security mechanism) 行で定義されている DCE や Kerberos などの他のセキュリティ・メカニズムとは異なります。master 行と query 行では、接続に使用されるセキュリティ・プロトコルを指定します。

たとえば、tli (トランスポート・レイヤ・インタフェース) と SSL を使用している UNIX マシンの一般的な *interfaces* ファイルは、次のようになります。

```
SERVER <retries><time-outs>

 query tli tcp /dev/tcp tli_add1 ssl
 master tli tcp /dev/tcp tli_add1 ssl
```

hostname にはクライアントが接続しているサーバの名前が、*address1* にはホスト・マシンのポート番号が入ります。

SSL フィルタを使用して、*interfaces* ファイルの master 行または query 行に指定されているサーバーに接続しようとする場合は、SSL プロトコルをサポートしている必要があります。サーバを、SSL 接続を受け入れ、他の接続によってプレイン・テキスト (非暗号化データ) を受け入れるように設定したり、他のセキュリティ・メカニズムを使用するように設定できます。

たとえば、SSL ベースの接続とプレイン・テキストの接続の両方をサポートする UNIX の Adaptive Server の *interfaces* ファイルは、次のようになります。

```
SYBSRV1
    master tli tcp /dev/tcp ¥x00020abc12345678000000000000000 ssl
    query tli tcp /dev/tcp ¥x00020abc12345678000000000000000 ssl
    master tli tcp /dev/tcp ¥x00020abd12345678000000000000000
```

UNIX の新しいスタイルの Sybase *interfaces* ファイルの場合は、次のようになります。

```
SYBSRV1
    master tli tcp hostname 2748 ssl
    query tli tcp hostname 2748 ssl
    master tli tcp hostname 2749
```

ソケットスタイルの *interfaces* ファイルでは、次のようになります。

```
SYBSRV1
    master tcp ether hostname 2748 ssl
    query tcp ether hostname 2748 ssl
    master tcp ether hostname 2749
```

これらの例では、SSL セキュリティ・サービスはポート番号 2748(0x0abc) に設定されています。SYBSRV1 では、Adaptive Server はポート番号 2749(0x0abd) でクリア・テキストを受信します。これには、セキュリティ・メカニズムやセキュリティ・フィルタがありません。

証明書によるサーバの検証

Open Client/Open Server が SSL 対応サーバに接続する場合は、サーバは証明書ファイルが必要です。これには、サーバの証明書と暗号化プライベート・キーが含まれています。また、証明書は認証局 (CA) がデジタル署名したものでなければなりません。

Open Client アプリケーションが Adaptive Server へのソケット接続を確立する方法は、既存のクライアント接続の確立に似ています。ネットワークのトランスポート層の接続コールがクライアント・サイドで完了し、受け入れ呼び出しがサーバ・サイドで完了すると、SSL ハンドシェイクが行われます。それから、ユーザのデータが送信されます。

SSL 対応サーバに正しく接続するには、次の手順に従ってください。

- クライアント・アプリケーションが接続要求を行った場合は、SSL 対応サーバは証明書を提出しなければなりません。
- クライアント・アプリケーションは、証明書に署名した CA を認識しなければなりません。「信頼された」CA すべてを含んだリストは、信頼されたルート・ファイルにあります。詳細については、次の「信頼されたルート・ファイル」を参照してください。
- SSL 対応サーバへの接続では、サーバ証明書の共通名は *interfaces* ファイルのサーバ名とも一致していなければなりません。

SSL 対応 Adaptive Server への接続を確立すると、Adaptive Server は起動時に *$SYBASE/$SYBASE_ASE/certificates/servername.crt* ディレクトリからサーバ自体のコード化された証明書ファイルをロードします。*servername* は、-S フラグを使用してサーバを起動するときにコマンド・ラインで指定したか、サーバの環境変数 $DSLISTEN に指定した Adaptive Server 名です。

他のタイプのサーバでは、別のロケーションに証明書を保管することがあります。サーバの証明書の保管場所については、ベンダ提供のマニュアルを参照してください。

信頼されたルート・ファイル

既知で信頼された CA のリストは、信頼されたルート・ファイルに保管されています。エンティティ (クライアント・アプリケーション、サーバ、ネットワーク・リソースなど) に既知で CA の証明書がある以外は、信頼されたルート・ファイルは証明書ファイルのフォーマットと同じです。システム・セキュリティ担当者が、標準 ASCII テキスト・エディタを使って CA を追加したり、削除したりします。

Open Client/Open Server の信頼されたルート・ファイルは *$SYBASE/config/trusted.txt* にあります。現時点で認識されている CA は、Thawte、Entrust、Baltimore、VeriSign、RSA です。

デフォルトでは、Adaptive Server はサーバ自身の信頼されたルート・ファイルを *$SYBASE/$SYBASE_ASE/certificates/servername.txt* に格納します。

Open Client と Open Server の両方を使用すると、次のように信頼されたルート・ファイルを別のロケーションに設定できます。

- Open Client

  ```
  ct_con_props (connection, CS_SET, CS_PROP_SSL_CA,
  "$SYBASE/config/trusted.txt", CS_NULLTERM, NULL);
  ```

  $SYBASE はインストール・ディレクトリが入ります。ct_config() を使用してコンテキスト・レベルに、または ct_con_props() を使用して接続レベルに CS_PROP_SSL_CA を設定できます。

- Open Server

  ```
  srv_props (context, CS_SET, SRV_S_CERT_AUTH,
  "$SYBASE/config/trusted.txt", CS_NULLTERM, NULL);
  ```

  $SYBASE にはインストール・ディレクトリが入ります。

サーバ証明書の取得

システム・セキュリティ担当者が、署名済みサーバ証明書とプライベート・キーをサーバにインストールします。次の手順によって、サーバ証明書を取得できます。

- 顧客環境に配備されている既存のパブリック・キー・インフラストラクチャで提供されているサードパーティのツールを使用します。
- Sybase 証明書要求ツールを信頼されたサードパーティの CA に使用します。

証明書を取得するには、CA に証明書を要求してください。サードパーティに証明書を要求し、その証明書が PKCS #12 フォーマットの場合は、certpk12 ユーティリティを使用して、Open Client/Open Server が理解できるフォーマットに証明書を変換します。詳細については、「certpk12 ユーティリティ」(113 ページ) を参照してください。

証明書要求ツールをテストし、認証方法がサーバで機能していることを確認するために、Open Client/Open Server は、検証目的で certreq ツールと certauth ツールを提供しています。このツールを使用すると、ユーザは認証局として機能できるので、認証局署名済み証明書をユーザ自身に発行できます。

サーバで使用する証明書を作成する主な手順は、次のとおりです。

1　証明書要求を生成します。

2　パブリック・キーとプライベート・キーのペアを生成します。

3　プライベート・キーを安全な場所に保管します。

4 証明書要求を認証局に送信します。

5 署名済みの証明書が CA から返信されたら、その証明書にプライベート・キーを付加します。

6 サーバのインストール・ディレクトリに証明書を格納します。

証明書を要求するサードパーティ・ツールの使用

多くのサードパーティ PKI ベンダといくつかのブラウザには、証明書とプライベート・キーを生成するユーティリティがあります。これらのユーティリティの多くはグラフィカルなウィザード形式で、一連の質問にユーザが答えると証明書の識別名と共通名が定義されます。

ウィザードに表示される指示に従って、証明書要求を作成してください。PKCS #12 フォーマットの署名済み証明書を受け取ったら、certpk12 を使用して、証明書ファイルとプライベート・キー・ファイルを生成します。2 つのファイルを *servername.crt* ファイルに連結します。*servername* はサーバの名前で、このファイルは、サーバのインストール・ディレクトリに配置されます。デフォルトでは、Adaptive Server の証明書は *$SYBASE/$SYBASE_ASE/certificates* に格納されます。詳細については、「certpk12 ユーティリティ」(113 ページ) を参照してください。

Sybase ツールによる証明書の要求と認証

Sybase では、証明書の要求と認証を行うツールを提供しています。certreq では、パブリック・キーとプライベート・キーのペア、および証明書要求を生成します。certauth では、*$SYBASE/$SYBASE_OCS/bin* ディレクトリでサーバ証明書要求を CA 署名済み証明書に変換します。

警告！ certauth は、テストにのみ使用してください。商用 CA のサービスを利用することをおすすめします。こうしたサービスではルート証明書の整合性が保護されており、広く承認された CA により署名された証明書を使用すれば、クライアント証明書を使用する形式の認証への移行が促進されるからです。

次の手順 1 ～ 5 に従って、サーバの信頼されたルート証明書を用意します。サーバ証明書を作成できることを確認するために、5 つの手順すべてを行い、検査用の信頼されたルート証明書を作成します。テスト版の認証局証明書 (信頼されたルート証明書) を作成した後で、手順 3 ～ 5 を繰り返してサーバ証明書に署名してください。

1 certreq を使用して、証明書を要求します。

2 certauth を使用して、証明書要求を CA の自己署名済み証明書 (信頼されたルート証明書) に変換します。

3 certreq を使用して、サーバ証明書とプライベート・キーを要求します。

4　certauth を使用して、証明書要求を CA 署名済みサーバ証明書に変換します。

5　プライベート・キーのテキストをサーバ証明書に追加し、サーバのインストール・ディレクトリに証明書を保存します。

これらの Sybase ツールの説明については、以下の項を参照してください。

注意 certauth と certreq は、RSA と DSA のアルゴリズムに依存しています。これらのツールは、ベンダが提供する暗号モジュールがある場合にのみ実行されます。この暗号モジュールでは、RSA と DSA アルゴリズムを使用して証明書要求を構築します。

Adaptive Server でサーバ証明書を追加、削除、表示する方法については、『システム管理ガイド』を参照してください。

Sybase ツールの説明

以下の項では、証明書の要求に使用できる Sybase ツールについて説明します。

certauth ユーティリティ

サーバ証明書要求を認証局 (CA) 署名済み証明書に変換します。

構文

certauth [-r] [-C] [-Q] [-K] [-O] [-P] [-T]
[-r]
[-C *caCert_file*]
[-Q *request_filename*]
[-K *caKey_filename*]
[-O *SignedCert_filename*]
[-P *caPassword*]
[-T *valid_time*]
or certauth -v

構文の説明を次に示します。

-r

テスト環境用の自己署名ルート証明書を作成します。

-C *caCert_file*

-r を指定したときに CA の証明書要求の名前を指定します。または、CA のルート証明書の名前を指定します。

-Q *request_filename*

証明書要求ファイルの名前を指定します。

-K *caKey_filename*

CA のプライベート・キーの名前を指定します。

-O *SignedCert_filename*

署名済み証明書ファイルを作成する場合に出力用に使用する名前を指定します。-r が指定されている場合、*SignedCert_filename* には自己署名ルート証明書が入ります。-r オプションを使用しない場合は、*SignedCert_filename* は *caCert_file* によって署名される証明書です。

-P *caPassword*

プライベート・キーの復号化に使用する CA のパスワードを指定します。

-T *valid_time*

署名済み証明書の有効期間を指定します。有効期間は日単位です。

-v

certauth のバージョン番号と版権メッセージを表示して、終了します。

この例では、CA の証明書要求 (*ca_req.txt*) を、プライベート・キー (*ca_pkey.txt*) を使用して証明書に変換します。プライベート・キーは *password* で保護されています。この例では、有効期間を 365 日に設定し、証明書に自己署名し、ルート証明書 (*trusted.txt*) として出力します。

```
certauth -r -C ca_req.txt -Q ca_req.txt
-K ca_pkey.txt -P password -T 365 -O trusted.txt
```

ユーティリティは、次のメッセージを返します。

```
 -- Sybase Test Certificate Authority --
Certificate Validity:
    startDate = Tue Sep 5  10:34:43  2000

    endDate = Wed Sep 5  10:34:43  2001

CA sign certificate SUCCEED (0)
```

注意 テスト CA 用に信頼されたルート証明書を 1 回だけ作成する必要があります。信頼済みルート証明書を作成してから、これを使ってテスト環境の多くのサーバ証明書に署名します。

この例では、サーバ証明書要求 (*srv5_req.txt*) を証明書に変換して、有効期間を 180 日に設定します。この例では、認証局の証明書 (*trusted.txt*) とプライベート・キー (*ca_pkey.txt*) を持つ証明書に署名し、パスワード保護を使用し、署名済み証明書を *sybase_srv5.crt* として出力します。

```
certauth -C trusted.txt -Q srv5_req.txt
-K ca_pkey.txt -P password -T 180 -O sybase_srv5.crt
```

注意 有効期間を設定しない場合は、デフォルトの 365 日が使用されます。

ユーティリティは、次のメッセージを返します。

```
 -- Sybase Test Certificate Authority --
Certificate Validity:
    startDate = Tue Sep  5 10:38:32  2000

    endDate = Sun Mar  4 09:38:32  2001

CA sign certificate SUCCEED (0)
```

次に、証明書の例を示します。サーバが使用できるサーバ証明書の作成手順については、次の「使用法」の項を参照してください。

```
-----BEGIN CERTIFICATE-----
MIICSTCCAgUCAVAwCwYHKoZIzjgEAwUAMG8xCzAJBgNVBAYTAlVTMRMwEQYDVQQI
 EwpDYWxpZm9ybmlhMRMwEQYDVQQHEwpFbWVyeXZpbGxlMQ8wDQYDVQQKFAZTeWh
c2UxDDAKBgNVBAsUA0RTVDEXMBUGA1UEAxQOc3liYXNlX3Rlc3RfY2EwHhcNMDAw
 ODE4MTkxMzM0WhcNMDEwODE4MTkxMzM0WjBvMQswCQYDVQQGEwJVUzETMBEGAUE
CBMKQ2FsaWZvcm5pYTETMBEGA1UEBxMKRW1lcnl2aWxsZTEPMA0GA1UEChQGU3li
 YXNlMQwwCgYDVQQLFANEU1QxFzAVBgNVBAMUDnN5YmFzZV90ZXN0X2NhMIHwMIo
BgcqhkjOOAQBMIGcAkEA+6xG7XCxiklxbP96nHBnQrTLTCjHlcy8QhIekwv9OlqG
EMG9AjJLxj6VCkPOD75vqVMEkaPPjoIbXEJEe/aYXQIVAPyvY1+B9phC2e2YFcf7
cReCcSNxAkBHt7rnOJZ1Dnd8iLQGt0wd1w4lo/Xx2OeZS4CJW0KVKkGId1hNGz8r
GrQTspWcwTh2rNGbXxlNXhAV5g4OCgrYA0MAAkA70uNEl90Kmhdt3RISiceCMgOf
1J8dgtWF15mcHeS8OmF9s/vqPAR5NkaVk7LJK6kk7QvXUBY+8LMOugpJf/TYMAsG
AhUAhM2Icn1pSavQtXFzXJUCoOmNLpkCFQDtE8RUGuo8ZdxnQtPu9uJDmoBiUQ==
-----END CERTIFICATE-----
```

使用法

- Adaptive Server が認識するサーバ証明書ファイルを作成するには、署名付き証明書ファイルの最後に証明書リクエスタのプライベート・キーを追加します。上記の例のように、署名済み証明書ファイル *sybase_srv5.crt* の最後に *srv5_pkey.txt* を貼り付けます。
- サーバが起動時にロードできる信頼されたルート・ファイルを作成するには、ファイル名 *trusted.txt* を *sybase_srv5.txt* に変更します。*sybase_srv5.txt* はサーバの共通名です。
- 次に、*sybase_srv5.txt* ファイルを Adaptive Server インストール・ディレクトリにコピーします。たとえば、*$SYBASE/$SYBASE_ASE/certificates* にコピーします。

SSL ベースのセッションに必要なファイルを、SSL 対応 Adaptive Server の起動に使用します。

作成した CA のルート証明書は、複数のサーバ証明書に署名するのに使用できます。

参照 certreq

certreq ユーティリティ

サーバ証明書要求と対応するプライベート・キーを作成する。このユーティリティは対話型モードで使用できます。また、コマンド・ラインにオプションのパラメータをすべて提供できます。

構文

certreq
[-F *input_file*]
[-R *request_filename*]
[-K *PK_filename*]
[-P *password*] または

certreq -v

パラメータ

-F *input_file*

属性情報のある入力ファイル名を指定して、証明書要求を構築します。*input_file* 名を指定しない場合は、必要な情報をユーザが対話形式で入力する必要があります。

input_file には、次のエントリが必要です。

```
req_certtype={Server,Client}
req_keytype={RSA,DSA}
req_keylength={for RSA: 512-2048;
     for DSA: 512,768,1024}
req_country={string}
req_state={string}
req_locality={string}
req_organization={string}
req_orgunit={string}
req_commonname={string}
```

注意 共通名はサーバ名と同じにしてください。

input_file と呼ばれるサンプル・ファイルについては、112 ページの例 2 を参照してください。

-R *request_filename*

証明書要求ファイルの名前を指定します。

-K *PK_filename*

プライベート・キー・ファイルの名前を指定します。

-P *password*

プライベート・キーを保護するために使用されるパスワードを指定します。

-v

バージョン番号と版権メッセージを表示して、終了します。

例 1

この例では、-F *input_file* パラメータを使用しないので、対話型モードになります。サーバ証明書要求 (*server_req.txt*) とプライベート・キー (*server_pkey.txt*) を作成するには、次のように入力します。

```
certreq

Choose certificate request type:
     S - Server certificate request
     C - Client certificate request (not supported)
     Q - Quit
Enter your request [Q] : s

Choose key type:

     R - RSA key pair
     D - DSA/DHE key pair
     Q - Quit
Enter your request [Q] : r

Enter key length (512, 768, 1024 for DSA; 512-2048 for RSA) :
512

Country: US

State: california

Locality: emeryville

Organization: sybase

Organizational Unit: dst

Common Name: server
```

ユーティリティから次のメッセージが返されます。

```
Generating key pair (please wait) . . .
```

キーのペアが生成された後、さらに情報を入力するためのプロンプトが certreq ユーティリティから表示されます。

```
Enter password for private key : password

Enter file path to save request: server_req.txt

Enter file path to save private key : server_pkey.txt
```

例 2

または、非対話型モードに -F オプションを使用することもできます。-F オプションを使用する場合は、有効値を使用し上記で説明したフォーマットに従ってください。これらに誤りがある場合、証明書は正しく作成されません。

次は、認証要求の非対話型エントリに使用できるサンプル・テキスト・ファイルです。

```
certreq -F input_file

req_certtype=server
req_keytype=RSA
req_keylength=512
req_country=us
req_state=california
req_locality=emeryville
req_organization=sybase
req_orgunit=dst
req_commonname=server
```

このファイルを作成、保存してから、コマンド・ラインに次のように入力します。

```
certreq -F path_and_file -R server_req.txt
-K server_pkey.txt -P password
```

ここでは、*path_and_file* には、テキスト・ファイルのロケーションが入ります。

このファイルは、サーバ証明書要求 (*server_req.txt*) とそのプライベート・キー (*server_pkey.txt*) を作成するものです。プライベート・キーは、*password* によって保護されます。

サーバ証明書ファイルは、標準的な ASCII テキスト・エディタを使用して編集できます。

使用法

- 入力ファイルでは、<tag>=*value* のフォーマットを使用します。<tag> は大文字と小文字を区別し、上記と同じでなければなりません。
- “=” は必須です。有効な *value* は、文字または数字で開始し、単一のワードにしてください。*value* の中には、スペースを入れないでください。
- *value* が必要な <tag> は、“req_certtype”、“req_keytype”、“req_keylength”、“req_commonname” です。
- <tag>、“=”、*value* の前後のスペースまたはタブは許容されます。ブランク行も許容されます。
- 各コメント行は、“#” で始めてください。
- 証明書要求ファイルは、PKCS #10 フォーマットになっています。これは certauth ツールで受け入れ可能な入力として使用されて、要求が CA 署名付きの証明書に変換されます。

参照

certauth

certpk12 ユーティリティ

PKCS #12 ファイルを証明書ファイルとプライベート・キーにエクスポートまたはインポートします。

構文

certpk12
{*-O Pkcs12_file* | *-I Pkcs12_file*}
[-C *Cert_file*]
[-K *Key_file*]
[-P *key_password*]
[-E *Pkcs12_password*]

certpk12 -v

パラメータ

-C *Cert_file*

-O が「オン」の場合は PKCS #12 ファイルにエクスポートする証明書ファイルの名前、-I が「オン」の場合は PKCS #12 ファイルからインポートする証明書ファイルの名前を指定します。

-K *Key_file*

-O がオンの場合は PKCS #12 ファイルにエクスポートするプライベート・キー・ファイルの名前、または -I がオンの場合は PKCS #12 ファイルからインポートするプライベート・キー・ファイルの名前を指定します。

-P *Key_password*

-K を指定しているプライベート・キーの保護に使用するパスワードを指定します。-O がオンの場合は、プライベート・キーを PKCS #12 ファイルにエクスポートするためのパスワードが必要です。-I がオンの場合は、PKCS #12 ファイルからプライベート・キーをインポートしてからテキスト・ファイルに出力するためのパスワードが必要です。

-O *Pkcs12_file*

エクスポートする PKCS #12 ファイルの名前を指定します。ファイルの内容は、証明書とプライベート・キー、証明書だけ、プライベート・キーだけの 3 つの場合があります。-O または -I のどちらかがオンになっていなければなりません。

-I *Pkcs12_file*

インポートする PKCS #12 ファイルの名前を指定します。ファイルの内容は、証明書とプライベート・キー、証明書だけ、プライベート・キーだけの 3 つの場合があります。-I または -O のどちらかがオンになっていなければなりません。

-E *Pkcs12_password*

PKCS #12 ファイルの保護に使用するパスワードを指定します。-O が「オン」の場合は、パスワードを使用してエクスポートする PKCS #12 ファイルを暗号化します。-I が「オン」の場合は、パスワードを使用してインポートする PKCS #12 ファイルを解読します。パスワードは「トランスポート・パスワード」とも呼ばれます。

-v

certpk12 ツールのバージョン番号と版権メッセージを表示して、終了します。

例 1

この例では、証明書ファイル *caRSA.crt* とプライベート・キー・ファイル *caRSApkey.txt* を PKCS #12 ファイル *caRSA.p12* にエクスポートします。*password* は *caRSApkey.txt* の解読に使用するパスワードです。*pk12password* は最終の *caRSA.p12* の暗号化に使用するパスワードです。

```
certpk12 -O caRSA.p12 -C caRSA.crt -K caRSApkey.txt
     -P password -E pk12password

-- Sybase PKCS #12 Conversion Utility certpk12 Thu Nov 9
16:55:51 2000--
```

例 2

この例では、証明書とプライベート・キーのある PKCS #12 ファイル *caRSA.p12* をインポートします。埋め込み証明書をテキスト・ファイル *caRSA_new.crt* に出力し、埋め込みプライベート・キーをテキスト・ファイル *caRSApkey_new.txt* に出力します。*new_password* を使用して *caRSApkey_new.txt* を保護し、*pk12password* を要求して *caRSA.p12* ファイルを解読します。

```
certpk12 -I caRSA.p12 -C caRSA_new.crt
  -K caRSApkey_new.txt  -P new_password -E pk12password

-- Sybase PKCS#12 Conversion Utility certpk12 Thu Nov 9
16:55:51 2000--
```

注意 例 1 と例 2 を実行すると、*caRSA.crt* と *caRSA_new.crt* は同じ内容になります。ただし、*caRSApkey.txt* と *caRSApkey_new.txt* はランダムに復号化されるので、同じにはなりません。

例 3

この例では、証明書ファイル *caRSA.crt* を PKCS#12 ファイル *caRSAcert.p12* にエクスポートします。*pkcs12password* を使用して *caRSAcert.p12* を暗号化します。

```
certpk12 -O caRSAcert.p12 -C caRSA.crt -E pk12password

-- Sybase PKCS#12 Conversion Utility certpk12 Thu Nov 9
16:55:51 2000--
```

例 4

この例では、証明書のある PKCS #12 ファイル *caRSAcert.p12* をインポートします。埋め込み証明書をテキスト・ファイル *caRSAcert.txt* に出力します。*pk12password* を要求して *caRSAcert.p12* ファイルを解読します。

```
certpk12 -I caRSAcert.p12 -C caRSAcert.txt
    -E pk12password

-- Sybase PKCS#12 Conversion Utility certpk12 Thu Nov 9
16:55:51 2000--
```

注意 例 3 と例 4 を実行すると、*caRSA.crt* と *caRSAcert.txt* は同じ内容になります。

使用法

- certpk12 がサポートしているのは、トリプル DES 暗号化方式により暗号化された PKCS #12 ファイルだけです。
- 証明書要求者のプライベート・キーを署名済み証明書ファイルの最後に付加します。
- ファイルに *servername.crt* と名前を付けます。*servername* はサーバの名前です。これを、*$SYBASE/$SYBASE_ASE* (Windows では *%SYBASE%¥%SYBASE_ASE%*) の下の証明書ディレクトリに置きます。

  SSL 対応 Adaptive Server の起動には、このファイルが必要です。

参照

certreq と certauth

索引

K

L

M

N

O