SYBASE®
An SAP® Company

設定ガイド

Open Client™/Open Server™

15.7

[UNIX 版]

ドキュメント ID：DC35838-01-1570-01
改訂：2012 年 6 月

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

目次

はじめに

『Open Client/Open Server 設定ガイド UNIX 版』では、Open Client™/Open Server™ を実行するためのシステム設定に関する情報について説明しています。Open Client/Open Server 製品を使用できるオペレーティング・システム・プラットフォームのリストについては、『新機能ノート Open Server および SDK Windows、Linux、UNIX 版』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、フェールオーバ時に 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・ユーティリティの使用方法について説明します。
- 「第 8 章 dsedit 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、dsedit ユーティリティの使用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 X-Windows ベースの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/Open Server 製品で使用する環境変数を列挙し、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
  - 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 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・サービスを列挙し、Open Client/Open 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 E Open Client/Open Server の SSL (Secure Socket Layer)」では、Open Client/Open Server の SSL (Security Sockets Layer) 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要約します。

関連マニュアル

詳細については、これらのマニュアルを参照できます。

- 『Open Server および SDK 新機能 Windows、Linux、UNIX 版』では、Open Server と Software Developer's Kit の新機能について説明しています。このマニュアルは、新機能の提供に伴って改訂されます。
- 使用するプラットフォームの『Open Server リリース・ノート』には、Open Server に関する重要な最新情報が記載されています。
- 使用するプラットフォームの『Software Developer's Kit リリース・ノート』には、Open Client™ および SDK に関する重要な最新情報が記載されています。
- 『jConnect™ for JDBC™ リリース・ノート』には、jConnect に関する重要な最新情報が記載されています。
- 『Open Client Client-Library/C プログラマーズ・ガイド』では、Client-Library アプリケーションの設計方法および実装方法について説明しています。
- 『Open Client Client-Library/C リファレンス・マニュアル』では、Open Client Client-Library™ のリファレンス情報について説明しています。
- 『Open Server Server-Library/C リファレンス・マニュアル』では、Open Server Server-Library のリファレンス情報について説明しています。
- 『Open Client および Open Server Common Libraries リファレンス・マニュアル』では、CS-Library のリファレンス情報について説明しています。CS-Library は、Client-Library と Server-Library の両方のアプリケーションで役に立つユーティリティ・ルーチンの集まりです。
- 『Open Server DB-Library/C リファレンス・マニュアル』では、Open Client DB-Library™ C バージョンのリファレンス情報について説明しています。
- 使用するプラットフォームの『Open Client/Server プログラマーズ・ガイド補足』では、Open Client/Server を使用するプログラマのために、プラットフォーム固有の情報について説明しています。このマニュアルには、次の情報が含まれています。
  - アプリケーションのコンパイルおよびリンク
  - Open Client/Server に含まれているサンプル・プログラム
  - プラットフォーム固有の動作をするルーチン

- 『Sybase SDK DB-Library Kerberos Authentication Option インストールガイドおよびリリース・ノート』では、DB-Library で使用される MIT Kerberos セキュリティ・メカニズムのインストールおよび有効化に関する情報について説明しています。DB-Library でサポートされる Kerberos セキュリティ・メカニズムの機能は、ネットワーク認証サービスと相互認証サービスのみです。
- 『Open Client Client-Library 移行ガイド』では、Open Client™ DB-Library™ のアプリケーションを Open Client Client-Library へ移行する方法について説明しています。
- 『Open Client/Server 開発者用国際化ガイド』では、国際化されたアプリケーションとローカライズされたアプリケーションを作成する方法について説明しています。
- 『Open Client Embedded SQL™/C プログラマーズ・ガイド』では、C アプリケーションで Embedded SQL および Embedded SQL プリコンパイラを使用する方法について説明しています。
- 『Open Client Embedded SQL™/COBOL プログラマーズ・ガイド』では、COBOL アプリケーションで Embedded SQL および Embedded SQL プリコンパイラを使用する方法について説明しています。
- 『jConnect for JDBC プログラマーズ・リファレンス』では、jConnect for JDBC 製品について説明し、リレーショナル・データベース管理システムに保管されているデータにアクセスする方法について説明しています。
- 『Sybase 製 Adaptive Server Enterprise ODBC ドライバ・ユーザーズ・ガイド』（Microsoft Windows および UNIX 版）では、Microsoft Windows および UNIX プラットフォームの Adaptive Server から、Open Database Connectivity (ODBC) ドライバを使用してデータにアクセスする方法について説明しています。
- 『Adaptive Server Enterprise データベース・ドライバ Perl プログラマーズ・ガイド』では、Perl 開発者向けに Perl スクリプトを使用して Adaptive Server データベースやクエリに接続する方法または情報を変更する方法について説明しています。
- 『Adaptive Server Enterprise 拡張モジュール PHP プログラマーズ・ガイド』では、PHP 開発者向けに Adaptive Server データベースに対してクエリを実行する方法について説明しています。

- 『Adaptive Server Enterprise 拡張モジュール Python プログラマーズ・ガイド』では、Adaptive Server データベースに対するクエリを実行するため使用可能な Sybase 固有の Python インタフェースの使用方法について説明しています。

その他の情報

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

- Getting Started CD には、リリース・ノートとインストール・ガイドが PDF 形式で含まれています。この CD は製品のソフトウェアに同梱されています。Getting Started CD に収録されているマニュアルを参照または印刷するには、Adobe Acrobat Reader が必要です（CD 内のリンクを使用して Adobe の Web サイトから無料でダウンロードできます）。
- Sybase Product Documentation Web サイトには、標準の Web ブラウザを使用してアクセスできます。また、製品ドキュメントのほか、EBFs/Maintenance、Technical Documents、Case Management、Solved Cases、ニュース・グループ、Sybase Developer Network へのリンクもあります。

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

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品認定の最新情報にアクセスする**

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

❖ **コンポーネント認定の最新情報にアクセスする**

1. Web ブラウザで Availability and Certification Reports (http://certification.sybase.com/) を指定します。
2. [Search By Base Product] で製品ファミリとベース製品を選択するか、[Search by Platform] でプラットフォームとベース製品を選択します。
3. [Search] をクリックして、入手状況と認定レポートを表示します。

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

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

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

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

Sybase EBF とソフトウェア・メンテナンス

❖ **EBF とソフトウェア・メンテナンスの最新情報にアクセスする**

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

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

3 製品を選択します。

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

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

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

表記規則

表 1：構文の表記規則

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

アクセシビリティ機能

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

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

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

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

不明な点があるときは

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

第 1 章

設定の概要

このマニュアルを読む前に、SDK および Open Server の『インストール・ガイド UNIX 版』の指示に従って、Open Client をインストールしてください。Open Client は、SDK (Software Developer's Kit) または Open Server ソフトウェアの一部としてこれらに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*、*ctlib*、Net-Library という名前の 3 つのアプリケーション・プログラミング・インタフェース (API) を提供します。これらの製品を使用することで、Adaptive Server Enterprise および Open Server アプリケーションと、カスタマ・アプリケーション、サード・パーティ製品、その他の Sybase 製品と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 以外）は、クライアントと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ルーチン
- クライアントとの対話をスケジュールするルーチン
- エラー条件を処理するルーチン
- サーバから要求されたその他の機能を実行するルーチン

Open Client/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/C リファレンス・マニュアル』
- 『Open Client DB-Library/C リファレンス・マニュアル』
- 『Open Server Server-Library/C 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/Open Server は、設定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。

  注意 アプリケーションが最新機能に確実にアクセスできるようにするには、バージョンを CS_CURRENT_VERSION に設定します。

- 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 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、DB-Library は 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/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します。

- ロケール固有 POSIX 環境変数 LC_*、LANG、LC_ALL、LC_COLLATE、*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します。
- *libtcl.cfg* ファイルを使用して、必要に応じてディレクトリ・ドライバとセキュリティ・ドライバをロードします。

接続プロセス

クライアントとサーバは「接続」を介して通信します。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は、アプリケーションの指定が優先されます。
- ディレクトリ・サービスまたは *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。
- ディレクトリ・サービスまたは *interfaces* ファイルを使用して、Open Server アプリケーションのアドレスを決定する。

注意　DSLISTEN を使用するのは、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だけです。

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、環境変数を設定します。Open Client/OpenServer 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します。
- 必要であれば、ネットワーク・ドライバを設定します。

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。

1 （DB-Library には適用されません）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 （DB-Library には適用されません）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。環境変数が定義されていない場合、Open Client はロケール名として "default" を使用します。

3 （DB-Library には適用されません）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 アプリケーションにターゲット・サーバが指定されていない場合は、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* 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* を調べて、前の手順で決定した名前と一致する SERVERNAME エントリを検出し、それに対応するターゲット・アドレスを使用します。

6　（DB-Library には適用されません）セキュリティ・サービス・ドライバの名前。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。

注意　項目 1-3 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。

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している必要があります。

LC_ALL または LANG を設定しない場合は、*locales.dat* の "default" エントリに、アプリケーションで使用するローカライゼーション値が反映されていることを確認します。

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。

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。

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。

4 ディレクトリ・ドライバまたはセキュリティ・ドライバを変更する場合は、*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 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。

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 LDAP 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 の使用方法については、「第 7 章 dscp の使用」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。

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。環境変数が定義されていない場合、Open Server はロケール名として "default" 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 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。

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・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 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。

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 LDAP 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 の使用方法については、「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」(72 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している必要があります。LC_ALL または LANG を設定しない場合は、*locales.dat* の "default" エントリに、アプリケーションで使用するローカライゼーション値が反映されていることを確認します。

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、DSQUERY 環境変数はターゲット・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については、Adaptive Server Enterprise の『高可用性システムにおける Sybase フェールオーバの使用』に記載されています。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 および Embedded SQL/COBOL は、バージョン 12.5.1 から 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 hafailover 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ー時に "MONEY1" に再接続できるようになっています。

```
PERSONNEL1
   master tcp ether huey 5000
   query tcp ether huey 5000
   hafailover MONEY1
```

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可能でなければなりません。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 および ct_con_props の各 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。このプロパティは、コンテキスト・レベルと接続レベルのどちらでも設定できます。次に、コンテキスト・レベルでプロパティを設定する例を示します。

```
CS_BOOL bhafailover = CS_TRUE;
retcode = ct_config(context, CS_SET, CS_HAFAILOVER,
&bhafailover, CS_UNUSED, NULL);
```

次に、接続レベルでのプロパティ設定を示します。

```
CS_BOOL bhafailover = CS_FALSE;
retcode = ct_con_props(connection, CS_SET,
CS_HAFAILOVER, &bhafailover, CS_UNUSED, NULL);
```

2　フェールオーバ・メッセージを処理します。コンパニオンがシャットダウン処理を始めると、クライアントはフェールオーバが発生するという情報メッセージを受け取ります。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います。

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 を発行しないと、フェールオーバ・プロパティ（たとえば isql -Q）が設定されたクライアントを接続できません。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、クライアントは sp_companion resume を発行するまでハングします。

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 hafailover 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、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できます。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。

LDAP サービスは、どのようなタイプでも（実際のサーバであっても、その他の LDAP サービスへのゲートウエイであっても)、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
Sybase インストール環境ごとに異なった構造	統一された階層構造
マスタ・エントリとクエリ・エントリが別々に存在する	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保存できない	サーバのメタデータを保存できる

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:1500
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 に、DIT ベースの有効な値を示します。

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。
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 • sybaseStatus の値 • アドレス：そのプロトコル・タイプに有効な任意のアドレス **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。

$SYBASE/$SYBASE_OCS/config ディレクトリに、次の LDAP サービスの LDAP ディレクトリ・スキーマが用意されています。

- *sybase.schema* － OpenLDAP サーバで使用するディレクトリ・スキーマが格納されている。
- *sybase-schema.conf* － Netscape 固有の構文を使用した、同じスキーマが格納されている。

- *sybase.ldf* － Microsoft Active Directory 用の Unicode フォーマットのディレクトリ・スキーマが格納されている。

前の例のエンティティは、"foobar" という名前の Adaptive Server がポート番号 5000 の TCP 接続で受信していることを示しています。また、このエンティティでは、12（回）のリトライ回数と 30（秒）のリトライ遅延時間も指定しています。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"）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。

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。 有効な値は次のとおり。 1 ― アクティブ 2 ― 停止 3 ― 失敗 4 ― 不明
Transport Address	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」(35 ページ) を参照。) OID の詳細については、「objectid.dat ファイル」(85 ページ) を参照。例については、*$SYBASE/config/objectid.dat* の [SECMECH] セクションを参照。

アプリケーションがディレクトリ・サービスを使用する仕組み

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_DS_PROVIDER) 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* 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は、Client-Library と Server-Library はフォールバックし、*interfaces* を使用して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 プロパティが ct_config で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() で設定されます。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。
- CS_LIBTCL_CFG プロパティは、代替の *libtcl.cfg* ファイルの名前とパスを指定します。
- *libtcl*.cfg* ファイルにパスワードとユーザ名のどちらも指定しない場合、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。「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、Sybase 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。

LDAP SDK ライブラリのロケーションと環境変数は、表 5-5 (28 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の、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ため、属性を設定する必要がありません。匿名アクセスは、一般には読み取り専用権限に使用します。
- ユーザ名とパスワード ― LDAP URL の拡張機能として *libtcl.cfg* ファイル（64 ビット・プラットフォームでは *libtcl64.cfg* ファイルで指定するか（「libtcl.cfg ファイルと libtcl64.cfg ファイル」(64 ページ) を参照)、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。

匿名アクセスの特徴は次のとおりです。

- 接続を確立するために、パスワードなどの認証情報を必要としません。
- 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 のプロパティで設定することもできます。*libtcl*.cfg* ファイル、および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。

❖ 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 (28 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、次のように変更します。

- *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 (20 ページ) を参照してください。

注意　LDAP URL は、1 行で記述してください。

このエントリのコンテキストは次のとおりです。

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

次に例を示します。

```
[DIRECTORY]
ldap=libsybdldap.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。

注意　64 ビット版のサポートには、上記の例の *lib3p* を *lib3p64* に、また *libsybdldap.so* を *libsybdldap64.so* に置き換えてください。

サポートされているプラットフォームについては、『Open Client Client-Library/C リファレンス・マニュアル』の「第 2 章 OpenLDAP」を参照してください。

表 5-4 に、*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_* プロパティ
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 必要なサード・パーティ・ライブラリが、適切な環境変数で指定されていることを確認します。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 HP-UX Itanium 32 ビット版	SHLIB_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
HP HP-UX Itanium 64 ビット版	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p64*
HP HP-UX PA-RISC 32 ビット版	SHLIB_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
HP HP-UX PA-RISC 64 ビット版	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p64*
Linux x86 32 ビット版	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
Linux x86-64 64 ビット版	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p64*
Linux POWER 32 ビット版	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
Linux POWER 64 ビット版	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p64*
Linux AMD64 (Opteron)/EM64T 版	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p64*
IBM AIX POWER 32 ビット版	LIBPATH	*$SYBASE/$SYBASE_OCS/lib3p*
IBM AIX POWER 64 ビット版	LIBPATH	*$SYBASE/$SYBASE_OCS/lib3p64*
Solaris x86-64 32 ビット版	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
Solaris x86-64 64 ビット版	LD_LIBRARY_PATH_64	*$SYBASE/$SYBASE_OCS/lib3p64*
Solaris SPARC 32 - ビット版	LD_LIBRARY_PATH	*$SYBASE/$SYBASE_OCS/lib3p*
Solaris SPARC 64 - ビット版	LD_LIBRARY_PATH_64	*$SYBASE/$SYBASE_OCS/lib3p64*

5 dscp または dsedit を使用して、LDAP サーバにサーバ・エントリを追加します。「サーバ・エントリの追加と変更」(41 ページ) と「ディレクトリ・サービスへのサーバの追加」(54 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=libsybdldap.so ldap://test:389/dc=sybase,dc=com
   ldap=libsybdldap.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、接続は *huey:11389* 上の LDAP サーバにフェールオーバします。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。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。

Microsoft Active Directory スキーマのインポート

ADAM インストール環境で提供されている ldifde.exe コマンドを使用して、*sybase.ldf* を Active Directory (AD) インスタンスまたは Active Directory Application Mode (ADAM) インスタンスにインポートできます。ディレクトリ・スキーマをインポートするには、次の構文を使用して ADAM インストール環境から ldifde.exe コマンドを実行します。

```
ldifde -i -u -f sybase.ldf -s server:port -b username
domain password -j .-c "cn=Configuration,dc=X"
#configurationNamingContext
```

Sybase サーバ・エントリ用のコンテナの作成

Active Directory にスキーマを正常にインポートしたら、Sybase サーバ・エントリ用のコンテナを作成し、コンテナと子オブジェクトに適切な読み込みと書き込みのパーミッションを設定します。

たとえば、相対識別名 (RDN) "CN=SybaseServers" をドメイン "mycompany.com" の Active Directory ルートに作成して、Sybase サーバ・エントリ名の保管と検索を行います。このコンテナのルート識別名 (rootDN) は、次のように *libtcl.cfg* ファイルに反映されます。

```
ldap=libsybdldap.dll ldap://localhost:389/
cn=SybaseServers,dc=mycompany,dc=com??...
```

Sybase サーバ・エントリの追加と修正を行うために、Active Directory にアカウント名 "Manager"、パスワード "secret" で専用のユーザ・アカウントを作成する場合、*libtcl.cfg* ファイルの完全なエントリは、次のようになります。

```
ldap=libsybdldap.so
ldap://myADhost:389/cn=SybaseServers,dc=mycompany,
dc=com????bindname=cn=Manager,cn=Users,dc=mycompay,
dc=com?secret
```

適切な読み込みと書き込みのパーミッションを設定したら、Sybase ユーティリティ・プログラム（dscp や dsedit など）を使用して、Active Directory 内の Sybase サーバ・エントリの保管、表示、修正を行うことができるようになります。

注意 Active Directory スキーマの拡張方法の詳細については、Microsoft Web サイトで「スキーマを拡張する」を検索してください。

SSL/TLS を使用した LDAP への接続

サポートされているすべてのプラットフォームで、SSL または TLS を使用して LDAP ディレクトリ・サーバへのセキュア接続を確立できます。クライアントと LDAP ディレクトリ・サーバとの間でセキュア接続を確立するには、次のいずれかの方法を使用します。

- *libtcl.cfg* ファイルに次の構文を入力して、LDAP サーバのセキュア・ポート（通常はポート番号 636）へのセキュア接続を確立します。

  ```
  [DIRECTORY]
  ldap=libsybdldap.so
  ldaps://huey:636/dc=sybase,dc=com????
  bindname=cn=Manager,dc=Sybase,dc=com?secret
  ```

  ldaps:// を使用してポート番号を指定しない場合、ポート番号 636 がデフォルトで使用されます。

- StartTLS を使用して、標準の接続（通常は、LDAP サーバのポート番号 389）をセキュア接続にアップグレードします。接続をアップグレードするには、*libtcl.cfg* ファイルに次の記述を入力します。

  ```
  [DIRECTORY]
  ldap=libsybdldap.so starttls
  ldap://huey:389/dc=sybase,dc=com????
  bindname=cn=Manager,dc=Sybase,dc=com?secret
  ```

  ldap:// を使用してポート番号を指定しない場合、ポート番号 389 がデフォルトで使用されます。

詳細については、『Open Client Client-Library/C リファレンス・マニュアル』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。

ネットワークベースの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、Kerberos セキュリティ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、サーバがサポートするセキュリティ・メカニズムを指定します。
- ディレクトリ・サービス・エントリのオプションの secmech 属性では、サーバがサポートするセキュリティ・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しているセキュリティ・メカニズムをサーバがサポートしているかどうかを確認できます。

- 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は、使用できるのはそれらのセキュリティ・メカニズムだけです。
- secmech 行や属性がない場合は、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、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 つの条件のどちらも満たされている必要があります。

- 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、Kerberos ドライバを使用するクライアントには Kerberos 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・サービスには、次のサービスが含まれています。

- 認証サービス
- パケット単位セキュリティ・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。

Kerberos 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 Kerberos セキュリティ・サービス」を参照してください。

アプリケーションがセキュリティ・サービスを使用する仕組み

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・トークン（Client-Library がサーバに送信する）と要求されたセキュリティ・サービスの種類に関する情報を返します。

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、サーバ・アプリケーションはログインを許可します。

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復号化と検証を実行します。

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・アプリケーションがセキュリティ・メカニズムを指定する場合、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・アプリケーションがセキュリティ・ドライバを指定しない場合、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・プロパ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読み込んで、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。

- Kerberos の設定
- libtcl.cfg 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。

Kerberos の設定

「付録 D Kerberos セキュリティ・サービス」および Kerberos のマニュアルを参照してください。

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。

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、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、dscp ユーティリティを使用して、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* ファイル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 LDAP 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* ファイルのサーバ・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。

次のコマンドを入力して、dscp を起動します。

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

dscp プロンプトの >> 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* 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* が指定されていない場合は、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*	サーバ・エントリ *SERVERNAME* を現在のセッションに追加する。 dscp は、*SERVERNAME* についての情報を要求します。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。
adtr *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します。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 *SESS* 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。

コマンド	説明
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する。

次のコマンドを入力する。

```
config
```

dscp ユーティリティは次の情報を画面に出力する。

- 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

ヘルプを参照する

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する。

```
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
```

SESS はセッション番号です。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 Entry Version	整数	15001	追加 ディレクトリ・サービス：なし *interfaces*：なし 変更 ディレクトリ・サービス：あり *interfaces*：なし
Server Name	文字列	N/A	追加 ディレクトリ・サービス：N/A *interfaces*：N/A 変更 ディレクトリ・サービス：なし *interfaces*: なし
Service	文字列	ASE	追加 ディレクトリ・サービス：あり *interfaces*：あり 変更 ディレクトリ・サービス：あり *interfaces*：なし
Server Status	整数	4 有効な値は次のとおり。 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：なし *interfaces*：なし 変更 ディレクトリ・サービス：あり *interfaces*：なし

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
Transport Address • トランスポート・タイプ • トランスポート・アドレス	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 有効なトランスポート・タイプの値は "tcp" 有効なトランスポート・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：あり トランスポート・アドレス：あり *interfaces*： トランスポート・タイプ：あり トランスポート・アドレス：あり
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：あり *interfaces*：あり 変更 ディレクトリ・サービス：あり *interfaces*：あり
HA Failoverserver (オプション)	文字列	なし	追加 ディレクトリ・サービス：あり *interfaces*：あり 変更 ディレクトリ・サービス：あり *interfaces*：あり

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、次のように入力します。

```
read SERVERNAME
```

たとえば、次のように入力します。

```
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:ASE
 Server Status:4 (Unknown)
 Server Address:
     Transport Type:tcp
     Transport Addr:victory 1824
     Transport Type:tcp
     Transport Addr:victory 1828
```

上記のサーバの属性については、表 7-2 を参照してください。

サーバ・エントリを追加する

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、次のように入力します。

```
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:[ASE]
 Transport Type:[tcp] tcp
 Transport Address:victory 8001
 Security Mechanism []:
```

add モードを終了するには、dscp プロンプト >> に戻るまで、[Enter] キーを押します。

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。

上記のサーバの属性については、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、*$SYBASE/$SYBASE_OCS/config/libtcl.cfg* ファイルを編集し、使用する LDAP サーバのエントリを追加して、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 サーバでログインにユーザ認証を要求する場合は、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 サーバのエントリにはアドレスのエントリを複数持つことができるので、もう一度 " トランスポート・タイプ " の入力を要求するプロンプトが表示されます。別のトランスポート・タイプを入力するか、フィールドは空白のまま [Enter] 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] キーを押します。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、Netscape または Mozilla ベースの Web ブラウザで以下の URL を入力します。

```
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・エントリを変更する

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、次のように入力します。

```
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:[ASE] 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 SERVERNAME
```

編集モードを終了するには、dscp プロンプト >> に戻るまで、[Enter] キーを押します。

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、次のように入力します。

```
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。次の処理を実行できます。

- 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。

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、次のように入力します。

```
copy myserver to my_server
```

dscp は、"myserver" と同じ内容の新しいエントリ "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。次の処理を実行できます。

- 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
```

各パラメータの説明は、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、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 to 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、次のように入力します。

```
copyall to 2
```

dscp は現在のセッションにあるすべての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

X-Windows ベース・グラフィカル・ツールの dsedit を使用すると、*interfaces* ファイルのサーバ・エントリを表示、編集できます。

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、*interfaces* のサーバ・エントリの設定には dscp または簡単なテキスト・エディタを使用します。詳細については、「第 7 章 dscp の使用」を参照してください。

dsedit の開始

サーバを追加または変更する場合は、*interfaces* 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。

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* ファイルの編集セッションを選択してオープンできます。

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」(55 ページ) を参照してください。

- ［Modify server entry］ー 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」(55 ページ) を参照してください。
- ［Copy server entry］ー 1 つ以上のエントリを別の *interfaces* ファイルにコピーします。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。

[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* ファイル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、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 を使用することをおすすめします。

dsedit を使用して、ディレクトリ・サービスと *interfaces* 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。「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 次を入力します。

- サーバ名 ― 必須。
- 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 の一覧は、*$SYBASE/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 度クリックして、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、サポートされる Web ブラウザまたは LDAP 管理ツールで次の URL を入力します。

```
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・エントリの追加、表示、編集

interfaces ファイルのサーバ・エントリを表示または編集するには、[Server Entry Editor] ウィンドウを使用します。[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、*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] ― アドレスのプロトコルおよびインタフェースを tcp などの値で指定します。
- 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] を選択し、TCP/IP アドレスを指定します。*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 Adaptive Server、またはバージョン 11.0.x 以前の Replication Server® の場合。
- 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動する、Open Client/Open Server® バージョン 12.0 以前の場合。

  注意 interfaces ファイル内での tli エントリは、Open Client/Open Server バージョン 12.5 から非推奨となっています。SDK と Open Server（DB-Library 含む）は tli フォーマットをサポートしますが、これの使用はおすすめしません。

- Solaris で、DB-Library が tcp 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。

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。

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か、誤ったディレクトリが指定されている。
- 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 が使用できない場合は、dsedit の代わりに dscp を使用します。

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* ファイルのパーミッションに関する問題がある。

  interfaces のエントリを編集するには、*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。

付 録　A

環境変数

この付録では、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	Sybase ホーム・ディレクトリのロケーション。 **注意**　CS_SYBASE_HOME プロパティは、代替の Sybase ホーム・ディレクトリの名前とパスを指定し、環境変数 *$SYBASE* を上書きします。	Open Client
SYBASE_OCS	Open Client/Open Server 製品のホーム・ディレクトリ。	*$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次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。

設定で使用する環境変数

Open Client/Open Server 製品は、設定プロセス中に表 A-2 に示す環境変数を使用します。

表 A-2: 設定で使用する環境変数

環境変数	説明	使用
SYBOCS_CFG	デフォルトの外部設定ファイル・パスの *$SYBASE/SYBASE_OCS/config/ocs.cfg* を上書きします。 詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。	ランタイム
SYBOCS_DBVERSION	実行時に DB-Library バージョン・レベルを外部から設定します。DB-Library は、DB-Library の初期化段階でこの変数を使用して環境変数を取得し、その環境変数値をバージョン・レベルとして保存する。 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。	ランタイム

環境変数	説明	使用
SYBOCS_DEBUG_FLAGS	特定の診断サブシステムを有効にする。複数のデバッグ・オプションを有効にするには、変数にカンマで区切ったフラグのリストを指定する。 デバッグの詳細については、『Open Client Client-Library C リファレンス・マニュアル』を参照。	ランタイム
SYBOCS_DEBUG_LOGFILE	診断を記録するログ・ファイルを指定する。この変数を設定しない場合、メッセージは stdout に書き込まれる。	ランタイム
SYBOCS_TCL_CFG	*libtcl.cfg* ファイルおよび *libtcl64.cfg* ファイルのフル・パス名を設定する。次に例を示す。 %setenv SYBOCS_TCL_CFG /usr/u/joe/libtcl.cfg	ランタイム

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; export VARIABLE
```

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; export DSQUERY
```

付 録　B

設定ファイル

この付録では、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。

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	参照先
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* **注意**　CS_LIBTCL_CFG プロパティまたは SYBOCS_TCL_CFG 環境変数を使用して *libtcl.cfg* ファイルへの代替パスを指定する。	「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。 『Open Client Client-Library/C リファレンス・マニュアル』も参照してください。
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* ファイルで記述されているサービスのバックアップとしても使用される。	*$SYBASE*	「interfaces ファイル」(72 ページ) を参照してください。

ファイル名	説明	ロケーション	参照先
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。	*$SYBASE/config/objectid.dat*	「付録 C ローカライゼーション」を参照してください。
ocs.cfg	ランタイム設定ファイルを使用すると、実行時に特定の値を変更できる。	*$SYBASE/$SYBASE_OCS/config*	「ocs.cfg ファイル」(76 ページ) を参照してください。

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（まとめて *libtcl*.cfg* ファイル）は、Open Client/Open Server 製品で使用する以下の 2 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

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。

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、いずれも 64 ビット・プラットフォーム上で提供されます。dsedit や srvbuild などの（64 ビット・プラットフォーム上の）32 ビット・アプリケーションは *libtcl.cfg* ファイルで設定情報を探し、64 ビット・アプリケーションは *libtcl64.cfg* ファイルで設定情報を探します。

libtcl.cfg* ファイルには、*interfaces* ファイルまたは LDAP ディレクトリ・サービスのどちらを使用するかを指定します。*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 -l 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 は、ディレクトリとセキュリティ・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 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、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。ドライバが見つからない場合には、手順 3 に進みます。

ライブラリのロケーションと環境変数は、表 5-5 (28 ページ) にリストされています。

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 の使用方法

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。

CS_LIBTCL_CFG 設定プロパティは、代替の *libtcl.cfg* ファイルの名前とパスを指定します。

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

libtcl.cfg ファイルは、ドライバのタイプごとに 2 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]

Open Client/Open Server のディレクトリ・サービスまたはセキュリティ・サービスのサポートを使用するには、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。

DIRECTORY セクション

[DIRECTORY] 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
```

各要素の意味は次のとおりです。

- *provider* はディレクトリ・サービスのローカル名です。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。LDAP ディレクトリ・ドライバは、次のようにプラットフォームに依存します。
  - HP HP-UX PA-RISC の場合は、*libsybdldap.sl* です。

    HP HP-UX Itanium、IBM AIX POWER、Solaris、Linux の各プラットフォームの場合は、*libsybdldap.so* です。
- *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。

DIRECTORY セクションの LDAP エントリ

最も簡単なフォームでは、LDAP ディレクトリ・サービスは、次のようなフォーマットで指定されます。

```
[DIRECTORY]
ldap=libsybdldap.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を許可している場合にだけ動作します。

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

LDAP URL への拡張機能として、*libtcl*.cfg* ファイルでユーザ名とパスワードを指定すると、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 ファイルと *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。

注意 LDAP URL は、1 行で記述してください。

```
ldap=libsybdldap.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=libsybdldap.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ている場合でも、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
```

各要素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/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」(85 ページ) を参照してください。

  Kerberos セキュリティ・メカニズムのデフォルトのローカル名は、"csfkrb5" です。デフォルト以外のローカル・メカニズム名を使用する場合は、オブジェクト ID ファイルにあるデフォルト名の後に、ローカル・メカニズム名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」(85 ページ) を参照）。

- *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
HP HP-UX PA-RISC 32 ビット版	Kerberos	*libsybskrb.sl*	CyberSafe TrustBroker 2.1 MIT Kerberos 1.4.1
HP HP-UX PA-RISC 64 ビット版	Kerberos	*libsybskrb64.sl*	MIT Kerberos 1.4.3
HP HP-UX Itanium 32 ビット版	Kerberos	*libsybskrb.so*	MIT Kerberos 1.4.1
HP HP-UX Itanium 64 ビット版	Kerberos	*libsybskrb64.so*	MIT Kerberos 1.4.1
IBM AIX POWER 32 ビット版	Kerberos	*libsybskrb.so*	CyberSafe TrustBroker 2.1 MIT Kerberos 1.4.1
IBM AIX POWER 64 ビット版	Kerberos	*libsybskrb64.so*	CyberSafe TrustBroker 2.1 MIT Kerberos 1.4.3
Linux x86 32 ビット版	Kerberos	*libsybskrb.so*	MIT Kerberos 1.4.1
Linux x86-64 64 ビット版	Kerberos	*libsybskrb64.so*	MIT Kerberos 1.4.1
Linux POWER 32 ビット版	Kerberos	*libsybskrb.so*	MIT Kerberos 1.4.1
Linux POWER 64 ビット版	Kerberos	*libsybskrb64.so*	MIT Kerberos 1.4.1
Solaris x86-64 32 ビット版	Kerberos	*libsybskrb.so*	MIT Kerberos 1.4.2
Solaris x86-64 64 ビット版	Kerberos	*libsybskrb64.so*	MIT Kerberos 1.4.2
Solaris SPARC 32 ビット版	Kerberos	*libsybskrb.so*	CyberSafe TrustBroker 2.1 MIT Kerberos 1.4.1
Solaris SPARC 64 ビット版	Kerberos	*libsybskrb64.so*	CyberSafe TrustBroker 2.1 MIT Kerberos 1.4.1

- *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。

  Kerberos ドライバの場合、*init-string* の構文は次のとおりです。

  ```
  secbase=@realm [libgss=<gss api V1 compatible
  library>]
  ```

  各要素の意味は次のとおりです。

  - *realm* は、デフォルトの Kerberos レルム名です。
  - （オプション）*libgss* は、GSS API バージョン 1 準拠ライブラリのフル・パスです。

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

- Kerberos

```
[SECURITY]
 csfkrb5=libsybskrb.so secbase=@ASE libgss=/krb5/lib/libgss.so
```

libgss=*/krb5/lib/libgss.so,* は、デフォルトの Kerberos レルムが Adaptive Server であり、ロードする GSS ライブラリが */krb5/lib/libgss.so* であることを意味します。

注意 GSS API ライブラリを指定する `libgss=<gss shared object path>` が使用される点に注意してください。複数のバージョンの Kerberos Client ライブラリが 1 台のマシンにインストールされている場合は特に、使用するライブラリのロケーションを明確に指定することが重要です。

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。この値は以下のプラットフォームによって異なります。

- IBM AIX POWER、Solaris および Linux の各プラットフォームおよび HP HP-UX Itanium には、*libsybdldap.so* を使用します。
- HP HP-UX PA-RISC には、*libsybdldap.sl* を使用します。

3　LDAP サーバのホストとポート番号を確認します。

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

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

LDAP 管理者はこの作業を行う必要がある場合があります。詳細については、LDAP のマニュアルを参照してください。

6　[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 ldap://host:port/ditbase
```

次に、LDAP ドライバの例を示します。

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

異なる DIT ベースを使用する複数の LDAP 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 や dsedit ツールを使用して LDAP 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]
ldap=libsybdldap.so ldap://lserv:389/dc=production,dc=sybase,dc=com
ldap1=libsybdldap.so ldap://lserv:389/dc=test,dc=sybase,dc=com
ldap2=libsybdldap.so ldap://backup1:389/dc=sybase,dc=com
```

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/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。Kerberos のデフォルト・ローカル名は *csfkrb5* です。

2 *driver* の値を指定します。この値はプラットフォームおよびセキュリティ・メカニズムによって異なります。(表 B-2 (69 ページ) はドライバ名のリストです。)

3 *init-string* の値を指定します。

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

```
secbase=@realmname [libgss=<gss api V1 compatible
library>]
```

各要素の意味は次のとおりです。

- *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。
- (オプション) *libgss* は、GSS API バージョン 1 準拠ライブラリのフル・パスです。

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。

```
csfkrb5=libsybskrb.so secbase=@ASE
libgss=/krb5/lib/libgss.so
```

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 *$SYBASE*/interfaces として作成されます。Open Client/Open Server 製品は、*$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 」、および『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 は *interfaces* エントリに標準フォーマットを使用します。

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>
```

各要素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* は、文字（ASCII の a-z、A-Z）で始まる必要があります。文字、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できます。
- *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” はクエリ行を指定します。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、まったく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、TCP/IP の場合 "tcp" です。
- *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。

  /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。有効な TCP/IP ポート番号の範囲は 1024 から 49151 までです。この範囲内にあるポート番号を使用することをおすすめします。

  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」(85 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 のようなオペレーティング・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、入力したアドレス文字列が正しくフォーマットされるので、作業が簡単になります。dscp を使用して *interfaces* ファイルを編集する詳細な手順については、「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に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* 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。

サーバの *interfaces* エントリには必要な数の代替サーバをリストできますが、各代替サーバは同一の *interfaces* ファイルにリストし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、*Client-Library* 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/C リファレンス・マニュアル』の「ランタイム設定ファイルの使い方」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。

この付録では、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 開発者用国際化ガイド』を参照してください。

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、アプリケーション実行時の特定の時点でローカライズする必要がある場合、Open Client/Open Server アプリケーションでは、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/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/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は、Open Client/Open Server は表 C-1 に示される環境変数も検索することがあります。

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

カスタム・ローカライゼーション時にアプリケーションが使用する環境変数については、『Open Client/Open Server 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。
  - ローカライズされたメッセージ・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

Open Client/Open Server アプリケーションは、実行時に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
  - Open Client/Open Server 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。
  - 以前のバージョンの 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。
  - システム管理ユーティリティ用のエラー・メッセージ・ファイルが入っている、*unicode* ディレクトリ。

- *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含まれています。
- *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。

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、プラットフォーム固有のロケール情報を 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、ロードするローカライゼーション情報を決定します。*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」(79 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 "default" 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]
```

各要素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ため、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、エントリが既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。

- "default" 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る。たとえば、次のように Sybase で事前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、LC_ALL 環境変数の値として "french" 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* に使用する任意の値を選択します。

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 *sortorder_name* の値を決定します（バイナリ以外のソート順が必要な場合）。

  charsets/*charset_name* サブディレクトリには、文字セットのソート順 (**.srt*) ファイルが入っています。*sortorder_name* は、これらのファイル名（*.srt* を除いた部分）のいずれかと一致する必要があります。

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) を必要に応じて更新します。

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、どのアプリケーションもそのエントリを使用していないことを確認してください。

ローカライズされたメッセージ・ファイル

警告！　ローカライズされたメッセージ・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含まれています。これらのメッセージ・ファイル（*locales/message/language_name* ディレクトリの **.loc* ファイル）を使用することで、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るようになります。

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含まれ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した場合、インストール・プロセスで *language_name* サブディレクトリが追加され、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

Open Client/Open Server のすべてのメッセージ・ファイルは、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、サポートされている各文字セットの照合順ファイルと、Unilib® が使用する変換ファイルが格納された *unicode* ディレクトリが入っています。

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 開発者用国際化ガイド』を参照してください。

Unicode 変換ファイル

Unicode 変換ファイルには、UTF-8 形式の Unicode (ISO 10646) 文字セットの変換設定情報が含まれています。これらの変換ファイルは、Sybase がサポートする各文字セットで利用できます。

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*) 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
```

各要素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]
        1.3.6.1.4.1.897.4.6.3 = 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。

付 録 D

Kerberos セキュリティ・サービス

この付録では、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、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-Library は Kerberos をサポートしません。

Open Client/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

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
- クレデンシャルの委任

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

- CyberSafe GSS ランタイム・ライブラリをインストールします。
- 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* または *libtcl64.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。
- アプリケーションに、サーバに接続するための既存のユーザ・クレデンシャルがあることを確認します。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。
- Client-Library アプリケーションを実行する前に、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ます。
- ユーザ名を入力する場合、ユーザ名はそのユーザの既存のクレデンシャルと一致する必要があります。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。
- 環境変数 CSFC5CCNAME は、クレデンシャル・キャッシュ・ファイル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。

  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、*libgss.so* または *libgss.sl* ファイルがパスに含まれていなければなりません。このファイルは 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。このファイルが 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。

- 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。

サンプル・プログラムの設定例と実行例については、*$SYBASE/$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。次に、サーバとそのクライアントで CyberSafe Kerberos セキュリティ・サービスを使用できるように、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（手順 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（Solaris および Linux プラットフォームの *libgss.so*、IBM AIX POWER の *libgss.so*、HP HP-UX の *libgss.sl*）は、使用しているプラットフォームの共有ライブラリ・パスで指定されたディレクトリに置く必要があります（表 5-5 (28 ページ) を参照してください)。または、*libtcl.cfg* の *libgss* キーワードを使用して、*GSS* ライブラリのパスを指定できます。

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リンシパル名を指定します。

Kerberos では、プログラムによる *key table* ファイルの指定が許可されていないため、CSFC5KTNAME 環境変数を使用する必要があります（手順 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」(34 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、ユーザ名はそのユーザの既存のクレデンシャルと一致する必要があります。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。

MIT Kerberos の設定

- MIT ソフトウェアをシステムにインストールし、設定します。使用しているプラットフォームでサポートされる MIT のバージョンについては、表 B-2 (69 ページ) を参照してください。
- ct_con_props を使用して必要なセキュリティ機能を設定するか、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。
- アプリケーションに、サーバに接続するための既存のユーザ・クレデンシャルがあることを確認します。つまり、アプリケーションのユーザは、kinit ユーティリティを使用して Kerberos 環境にログインしてから、クライアント・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、ユーザ名はそのユーザの既存のクレデンシャルと一致する必要があります。ユーザ名を入力しない場合、Client-Library はそのユーザの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。

- 環境変数 KRB5CCNAME は、クレデンシャル・キャッシュ・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。

  詳細については、マニュアルを参照してください。

- MIT GSS ライブラリの *libgssapi_krb5.so* は、libgss キーワードを使用して *libtcl.cfg* ファイルで指定する必要があります。Kerberos ドライバに関して、フル・パスを指定することをおすすめします。
- 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 Kerberos を設定したら、isql を使用して設定を検査できます。

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 MIT Kerberos

カスタム Open Server アプリケーションは Kerberos セキュリティで実行できます。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 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 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、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 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 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　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　KRB5_KTNAME 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（手順 2 を参照)。サーバ・キー・テーブル・ファイルが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、Kerberos 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　libgss キーワードを使用して、*libtcl.cfg* ディレクトリ内の *libgssapi_krb5.so* ファイルのロケーションを入力します。

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* ディレクトリ・サービスで定義されます。

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リンシパル名を指定します。

Kerberos では、プログラムによる *key table* ファイルの指定が許可されていないため、KRB5_KTNAME 環境変数を使用する必要があります（項目 4 を参照)。

Client-Library アプリケーションと MIT 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」(34 ページ) を参照してください。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、Kerberos にログインする必要があります。UNIX では、Kerberos kinit ユーティリティを使用して、Kerberos にログインしてください。
- ユーザ名を入力する場合、ユーザ名はそのユーザの既存のクレデンシャルと一致する必要があります。ユーザ名を入力しないと、Client-Library はそのユーザの Kerberos 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。

MIT Kerberos のクレデンシャル委任

Kerberos セキュリティ・ドライバは、MIT Kerberos GSS (Generic Security Services) ライブラリの使用時に、クレデンシャル委任をサポートしています。これにより、リモート・サーバとの接続を確立するときに、委任されたクライアント・クレデンシャルを使用する Open Server ゲートウェイ・アプリケーションを設定できます。

❖ **クレデンシャル委任を使用してリモート・サーバとの接続を確立するには**

これは、クレデンシャル委任の使用時に使用できる呼び出しシーケンスの一例です。ctos の例は *$SYBASE/OCS-15_0/sample/srvlibrary*. *connect.c* にあり、以下にあげるプロパティの例を含みます。

1 クライアント・アプリケーションは、次の構文を使用して、クレデンシャル委任を要求し、クレデンシャルをゲートウェイ接続に転送します。

```
ct_con_props(..., CS_SET, SRV_SEC_DELEGATION, ...)
```

2 ゲートウェイ・アプリケーションの接続ハンドラは、クライアントがクレデンシャル委任を要求しているかどうかをチェックします。

```
if (srv_thread_props(..., CS_GET,
     SRV_T_SEC_DELEGATION, ...))
    {...}
```

3 接続ハンドラは、委任されたクライアント・クレデンシャルを取得します。

```
srv_thread_props(..., CS_GET,
     SRV_T_SEC_DELEGCRED, ...)
```

4　クライアント・アプリケーションは、Client-Library 接続構造体内に、リモート・サーバへの接続に使用するための委任クレデンシャルを設定します。

```
ct_con_props(..., CS_SET, CS_SEC_CREDENTIALS, ...)
```

5　クライアント・アプリケーションは、ct_connect を使用してリモート・サーバへの接続を試みます。

isql と bcp オプション -Vd を使用して、クレデンシャル委任を要求することもできます。詳細については、『Open Client/Server プログラマーズ・ガイド補足 UNIX 版』を参照してください。

クレデンシャル委任の使用方法の詳細については、『Open Server Server-Library/C リファレンス・マニュアル』および『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。

Solaris Kerberos の設定

次に示す点を除き、Solaris Kerberos は MIT の Kerberos に基づいています。

- GSS ライブラリには、*libgssapi_krb5.so.* の代わりに */usr/lib/libgss.so* が使用されます。
- MIT Kerberos の設定に関するこの項で説明されている他の情報はすべて、Solaris で提供されているバージョンの Kerberos に当てはまります。

Kerberos 環境および混在 Kerberos 環境の設定

Kerberos 環境および混在 Kerberos 環境の設定については、Technical Document の General Kerberos Configuration Tasks (http://www.sybase.com/detail?id=1029260) を参照してください。

付 録 E

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

この付録では、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。

Open Client/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、以下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」

SSL の概要

SSL は、クライアントからサーバ、およびサーバからサーバへワイヤまたはソケット・レベルで暗号化されたデータを送信する業界標準です。サーバとクライアントは何度か I/O 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 対応サーバは証明書を提示し、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 CipherSuite のリストを返します。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 CipherSuite のリストについては、『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/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な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、*interfaces* ファイルの master 行と query 行のフィルタとして、SSL セキュリティ・メカニズムが設定されます。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 を使用します。

SSL フィルタは、*interfaces* ファイルの secmech (security mechanism) 行で指定されている 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、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>

  master tcp ether <hostname> <portnumber> ssl
  query tcp ether <hostname> <portnumber> ssl
```

hostname はクライアントが接続しているサーバの名前、*portnumber* はホスト・マシンのポート番号です。

interfaces ファイル内で SSL フィルタが指定されている master エントリまたは query エントリに接続するには、その接続で SSL プロトコルがサポートされ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ーン・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 tcp ether hostname 2748 ssl
    query tcp ether hostname 2748 ssl
    master tcp ether hostname 2749
```

この例では、SSL セキュリティ・サービスはポート番号 2748 に指定されています。SYBSRV1 では、Adaptive Server はポート番号 2749 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有効化

Open Client/Open Server が SSL 対応サーバに接続する場合は、サーバに証明書ファイルが必要です。このファイルは、サーバの証明書と暗号化されたプライベート・キーで構成されます。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。

既存のクライアント接続が確立されるのと同じように、Open Client アプリケーションは Adaptive Server へのソケット接続を確立します。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、受け入れコールがサーバ・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。「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、サーバ証明書内の共通名は *interfaces* ファイル内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 の名前です。

ほか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。サーバの証明書のロケーションの詳細については、ベンダ提供マニュアルを参照してください。

SDC 環境での共通名の検証

Open Client/Open Server における SSL 検証のデフォルトの動作は、サーバ証明書での共通名を ct_connect() で指定されたサーバ名と比較することです。共有ディスク・クラスタ (SDC：Shared Disk Cluster) 環境では、クライアントはサーバ名または SDC インスタンス名とは無関係の SSL 証明書の共通名を指定できます。クライアントは、複数のサーバ・インスタンスを表すクラスタ名で SDC に接続することも、特定の 1 つのサーバ・インスタンスに接続することもできます。

Open Client/Open Server は、SDC 環境での共通名の検証をサポートしています。このサポートにより、クライアントはトランスポート・アドレスを使用して、証明書の検証で使用される共通名を指定できるようになるため、Adaptive Server の SSL 証明書の共通名がサーバ名またはクラスタ名と異なっていてもかまいません。トランスポート・アドレスは、ディレクトリ・サービス (*interfaces* ファイル、LDAP、NT レジストリなど) のいずれか、または接続プロパティ CS_SERVERADDR で指定できます。

UNIX での構文

UNIX での SSL 対応 Adaptive Server およびクラスタのサーバ・エントリの構文を次に示します。

```
CLUSTERSSL
query tcp ether hostname1 5000 ssl="CN=name1"
query tcp ether hostname2 5000 ssl="CN=name2"
query tcp ether hostname3 5000 ssl="CN=name3"
query tcp ether hostname4 5000 ssl="CN=name4"

ASESSL1
master tcp ether hostname1 5000 ssl="CN=name1"
query tcp ether hostname1 5000 ssl="CN=name1"

ASESSL2
master tcp ether hostname2 5000 ssl="CN=name2"
query tcp ether hostname2 5000 ssl="CN=name2"

ASESSL3
master tcp ether hostname3 5000 ssl="CN=name3"
query tcp ether hostname3 5000 ssl="CN=name3"

ASESSL4
master tcp ether hostname1 5000 ssl="CN=name4"
query tcp ether hostname1 5000 ssl="CN=name4"
```

信頼されたルート・ファイル

信頼された既知の 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) に既知の 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・エディタを使って認証局を追加したり、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・ディレクトリです。ct_config() を使ってコンテキスト・レベルに、または ct_con_props() を使って接続レベルに 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 はインストール・ディレクトリです。

bcp ユーティリティと isql ユーティリティでも、別の場所にある信頼されたルート・ファイルを指定できます。パラメータ -x が構文に含まれており、このパラメータを使用して *trusted.txt* ファイルの場所を指定します。

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サードパーティの信頼済み CA に使用します。

証明書を取得するときは、CA の証明書を要求します。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。「certpk12 ユーティリティ」(111 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、ユーザが CA として機能し、CA- 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 CA に送信します。

5　署名付きの証明書が 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保管します。

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。

ウィザードの指示に従って、証明書要求を作成します。PKCS #12 フォーマットの署名付き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* はサーバの名前です。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。「certpk12 ユーティリティ」(111 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ためのツールを提供しています。certreq 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 は、サーバ証明書要求を CA の署名付き証明書に変換します。これらのツールは *$SYBASE/$SYBASE_OCS/bin* ディレクトリにあります。

警告！ certauth は、テスト専用で使用します。商用 CA のサービスを利用することをおすすめします。商用 CA はルート証明書の整合性を保護しており、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 CA 証明書 (信頼されたルート証明書) を作成したら、手順 3 ～ 5 を繰り返してサーバ証明書に署名します。

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 署名付き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付加して、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。

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、『ASE 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 *caCert_file*]
[-Q *request_filename*]
[-K *caKey_filename*]
[-N *serial_number*]
[-O *SignedCert_filename*]
[-P *caPassword*]
[-s *start_time*]
[-T *valid_time*]
[-v]

パラメータ

-r

テスト環境用の自己署名付きルート証明書を作成します。

-C *caCert_file*

-r を指定した場合は CA の証明書要求ファイル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。

-N *serial_number*

署名付き証明書のシリアル番号を指定します。-N が指定されていない場合、certauth は疑似ランダム・シリアル番号を生成します。

-O *SignedCert_filename*

署名付き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 を指定した場合、*SignedCert_filename* は自己署名付きルート証明書です。-r オプションを使用しない場合、*SignedCert_filename* は *caCert_file* によって署名された証明書です。

-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。

-s *start_time*

証明書の有効期間の開始時刻を指定します。有効期間は日単位で計算されます。-s を指定しなかった場合は、現在の時刻がデフォルトの開始時刻となります。

-T *valid_time*

証明書の有効期間を指定します。有効期間は日単位で計算されます。

-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、プライベート・キー (*ca_pkey.txt*) を使用して、CA の証明書要求 (*ca_req.txt)* を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。信頼されたルート証明書を作成したら、この証明書を使用して、テスト環境の多くのサーバ証明書に署名できます。

この例では、サーバ証明書要求 (*srv5_req.txt*) を証明書に変換し、有効期間を 180 日に設定します。ここでは、CA 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、署名付き証明書を *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

- -N オプションで指定するシリアル番号の最大長は、16 進文字で 20 文字です。指定したシリアル番号がこれよりも長い場合、certauth はシリアル番号を最大長にトランケートします。

- 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。

CA のルート証明書を作成したら、この証明書を使用して、複数のサーバ証明書に署名できます。

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・キーを作成します。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*]
[-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します。

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 複数のサーバが同じ共通名を使用できるクラスタ環境以外では、共通名はサーバ名と同じ名前にしてください。

詳細については、「SDC 環境での共通名の検証」(100 ページ) を参照してください。

サンプル・ファイルの *input_file* については、*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。この証明書要求ファイルは、certauth ツールが要求を CA の署名付き証明書に変換するときに受け入れ可能な入力として使用されます。

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*]
[-v]

パラメータ

-C *Cert_file*

-O をオンにしている場合は、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を指定します。-I をオンにしている場合は、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 がオンの場合には、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 がオンの場合には、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 ファイルを保護するために使用するパスワードを指定します。-O をオンにしている場合は、エクスポートする PKCS #12 ファイルを暗号化するときにパスワードを使用します。-I をオンにしている場合は、インポートする PKCS #12 ファイルを復号化するときにパスワードを使用します。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* の復号化に使用するパスワードです。*pk12password* は、最後の *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 2009--
```

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* は、*caRSApkey_new.txt* を保護するために使用し、*pk12password* は、*caRSA.p12* ファイルの復号化に必要です。

```
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 2009--
```

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* は、*caRSAcert.p12* の暗号化に使用します。

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

-- Sybase PKCS#12 Conversion Utility certpk12 Thu Nov 9
16:55:51 2009--
```

例 4

この例では、証明書を含む 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* は、*caRSAcert.p12* ファイルの復号化に必要です。

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

-- Sybase PKCS#12 Conversion Utility certpk12 Thu Nov 9
16:55:51 2009--
```

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 暗号化方式で暗号化された PKCS #12 ファイルだけです。
- 証明書要求者のプライベート・キーを署名付き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* の下の証明書ディレクトリに置きます。

  このファイルは、SSL が有効な Adaptive Server を起動するときに必要です。

参照

certreq と certauth

パスワード暗号化のための FIPS 140-2 準拠

Open Client と Open Server のログイン・パスワードとリモート・パスワードの暗号化は、Sybase CSI (Common Security Infrastructure) によって実現されます。Certicom SSL Plus 5.2.2 CSI-Crypto 2.6 は、連邦情報処理標準 (FIPS: Federal Information Processing Standard) 140-2 に準拠しています。この機能をサポートする UNIX プラットフォームでは、CSI 2.6 とともにインストールされる、*libsbgse2.so* という名前の Certicom Security Builder 共有ライブラリが必要です。FIPS 暗号化をサポートするために、SDK または Open Server のインストール時に、*libsbgse2.so* という名前の Certicom Security Builder 共有ライブラリが *$SYBASE/$SYBASE_OCS/lib3p* または *$SYBASE/$SYBASE_OCS/lib3p64* にインストールされます。

索引

L

M

N

O

P

S

U

あ

か

き

け

さ

し

せ

そ

て

と

ね

は

へ

ほ

ろ