SYBASE®

開発者用国際化ガイド

Open Client™/Open Server™

15.5

ドキュメント ID：DC30524-01-1550-01
改訂：2009 年 11 月

マニュアルの注文

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

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

目次

はじめに

対象読者

このマニュアルは、Open Client/Server アプリケーション開発者を対象としています。Client-Library™、DB-Library™、Embedded SQL™、および Server-Library に関する基礎知識を持っていることを前提としています。

このマニュアルの内容

このマニュアルには、以下の章があります。

- 「第 1 章 国際化とローカライゼーションの概要」では、国際化とローカライゼーションについて定義するとともに、国際化されたアプリケーションを開発した場合の利点について説明します。
- 「第 2 章 ローカライゼーションの機能」では、Open Client/Server のローカライゼーション・メカニズムの機能について説明します。
- 「第 3 章 国際化された Open Client/Server アプリケーションの開発」では、国際化された Open Client/Server アプリケーションの開発方法について説明します。
- 「第 4 章 コード化文字セット変換のサポート」では、Open Client/Server 製品で文字セット変換がどのように行われるかについて説明します。
- 「第 5 章 ロケール・ファイルの編集」では、ロケール・ファイルの概要とその変更方法について説明します。
- 「第 6 章 照合順の作成または変更」では、照合順ファイルの作成方法と変更方法について説明します。
- 「付録 A 国際化に関連するディレクトリとファイル」では、国際化に関係のある Open Client/Server のディレクトリとファイルについて説明します。
- 「付録 B 外部ローカライゼーション・ファイルの構文」では、外部ローカライゼーション・ファイルの構文について説明します。

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 リリース・ノート Microsoft Windows 版』には、Open Server に関する重要な最新情報が記載されています。
- 『Software Developer's Kit リリース・ノート Microsoft Windows 版』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 バージョン 6.05 および 7.0』には、jConnect™ に関する重要な最新情報が記載されています。

- この『Open Client/Server 設定ガイド Microsoft Windows 版』では、システムを設定して Open/Client Server 製品を実行する方法について説明します。
- 『Open Client Client-Library/C リファレンス・マニュアル』では、Open Client Client-Library のリファレンス情報について説明しています。
- 『Open Client 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 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 Client/Server プログラマーズ・ガイド補足 Microsoft Windows 版』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
- 『jConnect for JDBC インストール・ガイド バージョン 6.05』では、jConnect for JDBC™ のインストール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。
- 『Adaptive Server Enterprise ADO.NET Data Provider ユーザーズ・ガイド』では、C#、Visual Basic .NET、マネージ拡張を備えた C++、J# など、.NET でサポートされる任意の言語を使用して Adaptive Server® 内のデータにアクセスする方法について説明しています。
- 『Sybase 製 Adaptive Server Enterprise ODBC ドライバのユーザーズ・ガイド』(Windows および Linux 版) では、Windows、Linux、および Apple Mac OS X プラットフォームの Adaptive Server から、Open Database Connectivity (ODBC) ドライバを使用してデータにアクセスする方法について説明します。
- 『Sybase 製 Adaptive Server Enterprise OLE DB プロバイダのユーザーズ・ガイド Microsoft Windows 版』では、Microsoft Windows プラットフォームの Adaptive Server から、OLE DB プロバイダを使用してデータにアクセスする方法について説明します。

その他の情報

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

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

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

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

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

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品認定の最新情報にアクセスする**

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

❖ **コンポーネント認定の最新情報にアクセスする**

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

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

3 [Search] をクリックして、入手状況と認定レポートを表示します。

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

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

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

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

Sybase EBF とソフトウェア・メンテナンス

❖ **EBF とソフトウェア・メンテナンスの最新情報にアクセスする**

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

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

3 製品を選択します。

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

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

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

表記規則

表 *1:* 構文の表記規則

キー	定義
command	コマンド名、コマンドのオプション名、ユーティリティ名、ユーティリティのフラグ、キーワードは sans serif で示す。
variable	変数 (ユーザが入力する値を表す語) は斜体で表記する。
{ }	中カッコは、その中から必ず 1 つ以上のオプションを選択しなければならないことを意味する。コマンドには中カッコは入力しない。
[ ]	角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには中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

国際化とローカライゼーションの概要

この章では、国際化とローカライゼーションについて定義するとともに、国際化されたアプリケーションを開発した場合の利点について説明します。

この章の内容は、次のとおりです。

国際化とローカライゼーション

「国際化」とは、アプリケーションを複数の言語や文化的慣例に対応させることをいいます。

国際化されたアプリケーションでは、実行時に外部ファイルを使用して、言語固有の情報を表示します。このようなアプリケーションは、言語固有のコードを含んでいないので、コードに変更を加えることなくどのようなネイティブ言語の環境にも配備できます。

「ローカライゼーション」とは、特定の言語とその言語に関連する文化的慣例 (日時表現など) を使用して実行するようにアプリケーションを設定する処理です。

ローカライズされたアプリケーションでは、配備するネイティブ言語環境の見た目と操作性が適用されます。ローカライズされたアプリケーションはローカルの言語と文字セットでメッセージを生成し、日付と時刻にはローカルの表現規則を使用します。

Open Client/Server 製品では、柔軟性があって強力なローカライゼーション・メカニズムが提供されます。このため、アプリケーション・プログラマは国際化アプリケーションを設計および開発できます。

国際化されたアプリケーションの利点

アプリケーションを他国でも使用できるように設計することは、きわめて面倒な仕事に思えます。プログラマたちは、国際化とは文化や言語上の慣習によって必要となる部分を個々にハードコードすることだと考えがちです。

しかし、もっと良いアプローチがあります。それは、国際化されたアプリケーション、つまり実行時にローカル・コンピューティング環境を調べて使用する言語を決定し、その言語に関する情報が記述されているファイルを読み込むアプリケーションを作成することです。

国際化されたアプリケーションであれば、同じアプリケーションをどの国でも使用できます。このアプローチには次のような利点があります。

- アプリケーションを 1 つだけ作成すればよく、各国語版を個々に作成する必要はありません。
- アプリケーションの提供先の国が増えたときも、アプリケーションに変更を加える必要はありません。その国のローカライゼーション・ファイルを添付するだけで済みます。
- すべてのサイトで機能と動作を統一できます。

国際化システム

国際化システムでは、国際化されたクライアント・アプリケーション、ゲートウェイ、サーバをさまざまなネイティブ言語環境の複数のプラットフォームで実行できます。

たとえば、次のようなコンポーネントで構成される国際化システムもあります。

- ニューヨーク、メキシコシティ、パリにある発注処理アプリケーション (Client-Library アプリケーション)
- ドイツにある在庫管理サーバ (Adaptive Server® Enterprise)
- フランスにある発注遂行アプリケーション (Adaptive Server Enterprise)
- 日本にある中央会計アプリケーション (Adaptive Server Enterprise と相互稼働する Open Server アプリケーション)

このシステムの発注処理アプリケーションは、以下の処理を実行します。

- 在庫管理サーバに対してクエリを発行し、注文された商品の在庫があるかどうかを調べる。
- 発注遂行サーバに発注データを送る。
- 財務情報を会計アプリケーションに送る。

在庫管理サーバと発注遂行サーバはクエリに応答し、会計アプリケーションは財務データを収集してレポートを作成します。

すべてのアプリケーションとサーバはローカルの言語と文字セットを使用して、入力の受け入れ、メッセージの生成を行います。

上記のシステムでは、発注処理アプリケーションと Open Server ゲートウェイは、ロケール名を指定する LC_ALL 環境変数を使用してローカライズされます。Open Client/Server アプリケーションは、実行時に指定されたロケール名と Sybase ロケール・ファイル内のエントリが一致しているかどうかを調べて、ロードする言語、文字セット、照合順ファイルを決定します。

このシステムの Adaptive Server Enterprise は、サーバとともにインストールされた言語モジュールを使用してローカライズされています。

Open Client/Server での国際化システムのサポート

Open Client/Server 製品には、国際化システムの開発を完全にサポートする機能が備わっています。Client-Library、Server-Library、CS-Library を使用すればサポートされているどのようなプラットフォームにもローカライズできるので、アプリケーションは以下を扱うことができます。

- エラー・メッセージ用の特定の言語と文字セット
- 文字列を別の文字セットから変換するときの特定の文字セット
- 文字列のソートまたは比較を行うときに使用する特定の照合順
- 特定の日時フォーマットと値

注意　DB-Library は、エラー・メッセージを処理するために一度に 1 つの言語と文字セットをサポートします。詳細については、「国際化された DB-Library アプリケーションの開発」(30 ページ) を参照してください。

Adaptive Server Enterprise アプリケーションと Open Server アプリケーションはともに、ローカライズされた Open Client アプリケーションをサポートします。クライアントがサーバに接続する場合、サーバは必要な文字セット変換がある場合にはその変換をサポートできるかどうかを調べます。

Open Client と Open Server が Unicode 標準をサポートしているので、Open Server アプリケーションは、使用する文字セットに関係なくどのようなクライアントでもサポートできます。

Adaptive Server Enterprise 12.5 以降が Unicode をサポートしています。以前のバージョンの Adaptive Server Enterprise の文字セット変換を実行する Open Server アプリケーションを使用できます。「変換ゲートウェイとしての Open Server の使用」(41 ページ) を参照してください。

第 2 章

ローカライゼーションの機能

この章では、Open Client/Server のローカライゼーション・メカニズムがどのように機能するかについて説明します。

この章の内容は、次のとおりです。

注意 この章の内容は DB-Library には当てはまりません。

使用するローカライゼーション値の決定

アプリケーションをどのようにローカライズするか、つまり、どの環境でどの言語、文字セット、文化的慣例を使用するかを決定してから、国際化された Open Client/Server アプリケーションを開発する必要があります。

Open Client/Server アプリケーションは「初期ローカライゼーション値」または「カスタム・ローカライゼーション値」、あるいはその両方を使用できます。

- アプリケーションが (cs_ctx_alloc を呼び出して) コンテキスト構造体を割り付ける場合、初期ローカライゼーション値は実行時に次のように決定されます。
  - LC_ALL 環境変数を設定した場合、アプリケーションはその値を使用して新しいコンテキスト構造体をローカライズします。
  - LC_ALL 環境変数を設定しないで LANG 環境変数を設定した場合、アプリケーションはその値を使用して新しいコンテキスト構造体をローカライズします。
  - 上記のどちらの環境変数も設定しない場合、アプリケーションはロケール・ファイル内の “default” エントリを使用して、新しいコンテキスト構造体をローカライズします。ロケール・ファイル (*locales.dat*) は以下のディレクトリにあります。

    - UNIX プラットフォームの *$SYBASE/locales* ディレクトリ
    - *Windows* の *%SYBASE%¥locales* ディレクトリ
- アプリケーションは CS_LOCALE 構造体内に必要な情報を提供する cs_locale を呼び出してカスタム・ローカライゼーション値を設定してから、この CS_LOCALE 構造体を使用して、コンテキスト、接続、スレッド、データ要素、またはルーチンのローカライゼーション値を変更します。

初期ローカライゼーション値の使用

通常、国際化された Open Client/Server アプリケーションは、LC_ALL、LANG、または *locales.dat* ファイル内の “default” エントリによって特定された初期ローカライゼーション値を使ってローカライズが行われます。

Open Client/Server アプリケーションが CS-Library ルーチン cs_ctx_alloc を呼び出して CS_CONTEXT 構造体を割り付ける場合、初期ローカライゼーション値は実行時に決定されます。アプリケーションがこの呼び出しを行った場合、CS-Library は新しいコンテキスト構造体に初期ローカライゼーション情報をロードします。

ローカライゼーション情報の内容は次のとおりです。

- 言語
- 文字セット
- 照合順
- 日付と時刻のフォーマット

ロード処理は次の手順で行われます。

1　アプリケーションが cs_ctx_alloc を呼び出します。

2　CS-Library はロケール名を決定するために LC_ALL 環境変数または LANG 環境変数の環境を検索します。表 2-1 ではこの検索について説明します。

表 *2-1: CS-Library* がロケール名を決定する方法

LC_ALL が定義されているか？	LANG が定義されているか？	CS-Library のアクション
はい	N/A	ロケール名として LC_ALL 環境変数の値を使用する。
いいえ	はい	ロケール名として LANG 環境変数の値を使用する。
いいえ	いいえ	“default” のロケール名を使用する。これは、CS-Library が次のどちらかをロードすることを意味する。 • 出荷時にプラットフォームごとに設定されているデフォルト • “default” のロケール名に割り当てられたユーザ定義セット

3　CS-Library は *locales.dat* ファイル内でロケール名を検索し、対応する言語と文字セットを決定します (照合順が指定される場合もあれば、指定されない場合もあります)。*locales.dat* ファイル内にロケール名がない場合は、cs_ctx_alloc によってエラーが返されます。

4　CS-Library は、適切なローカライゼーション情報を使用して新しいコンテキスト構造体をロードします。

初期ローカライゼーション値を使用するアプリケーションの設定

アプリケーションが初期ローカライゼーション値を使用する場合は、アプリケーションを国際化するために特別なコードを指定しないでください。ただし、システム管理者とユーザがアプリケーションの環境変数の設定方法を理解している必要があります。

アプリケーションを分散する場合は、システム管理者とユーザは次の点を理解しておく必要があります。

- LC_ALL 環境変数が存在している場合、その値は *locales.dat* ファイル内の正しいエントリと一致している必要があります。
- LANG 環境変数が存在している場合、その値は *locales.dat* ファイル内の正しいエントリと一致している必要があります。

- 上記のどちらの環境変数も存在していない場合、*locales.dat* ファイル内の “default” エントリが正しいエントリになっている必要があります (つまり、アプリケーションが使用する言語、文字セット、照合順がリストされる必要があります)。

カスタム・ローカライゼーション値の使用

Client-Library アプリケーションと Open Server アプリケーションは、コンテキスト・レベル、接続レベル、スレッド・レベル、データ要素レベル、ルーチン・レベルでカスタム・ローカライゼーション値を使用できます。

Client-Library アプリケーションまたは Open Server アプリケーションは、次の手順に従ってカスタム・ローカライゼーション値を設定します。

1 cs_locale を呼び出し、特定のローカライゼーション値を使用して CS_LOCALE 構造体をロードします。「cs_locale ルーチン」(12 ページ) を参照してください。

2 ロードされた CS_LOCALE 構造体を使用して、コンテキスト、接続、スレッド、またはデータ要素をカスタマイズします。この処理の詳細については、「CS_LOCALE 構造体」(11 ページ) を参照してください。

コマンド・ライン・オプションを指定し、カスタム・ローカライゼーション値を使用して Embedded SQL プリコンパイラを実行できます。

Embedded SQL アプリケーションはカスタム値を使用できません。つまり、LC_ALL 環境変数、LANG 環境変数、または *locales.dat* ファイル内の “default” エントリによって実行時に決定される初期ローカライゼーション値は変更できません。

ローカライゼーション・メカニズムの詳細

この項では、ローカライゼーション・メカニズムについてさらに詳しく説明します。具体的には *locales.dat* ファイル、ローカライゼーション環境変数、CS_LOCALE 構造体、cs_locale ルーチンについて説明します。

ロケール・ファイル

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

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

ロケール・ファイルの詳細については、「第 5 章 ロケール・ファイルの編集」を参照してください。

環境変数

大部分のプラットフォームでは、Client-Library アプリケーションと Server-Library アプリケーションは次のようなローカライゼーション環境変数を使用します。

- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGE
- LC_TIME
- LANG

注意　ユーザのログイン時に、ローカライゼーション環境変数が特定の値に自動的に設定されるシステム (通常 UNIX システム) もあります。ご使用のシステムがこれに該当する場合は、変数の値を *locales.dat* ファイルのロケール名に一致させるか、またはログオン後に変数をリセットします。

表 2-2 は、Open Client/Server アプリケーションがこれらの環境変数をどのように使用するかを示します。

表 2-2: ローカライゼーションに関係のある環境変数

環境変数	使用目的	使用箇所	参照箇所
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。	Client-Library アプリケーションまたは Open Server アプリケーション	アプリケーションが cs_ctx_alloc または cs_ctx_global を呼び出す場合に参照される。 アプリケーションが *type* パラメータに CS_LC_ALL、*buffer* パラメータに NULL を指定して cs_locale を呼び出す場合に参照される。
		Embedded SQL プリコンパイラ	アプリケーションのプリコンパイル時に、プリコンパイラ・メッセージに使用するデフォルトの言語と文字セットを決定する場合に参照される。
		プリコンパイルされた Embedded SQL アプリケーション	アプリケーションの実行時に、プリコンパイルされたアプリケーションが初めて cs_ctx_global を呼び出す場合に参照される。 プリコンパイラはそれぞれの Embedded SQL 文に対して cs_ctx_global 呼び出しを生成する。
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)	Client-Library アプリケーションまたは Open Server アプリケーション	アプリケーションが *type* パラメータに CS_LC_COLLATE を指定し、*buffer* パラメータに NULL を指定して cs_locale を呼び出す場合に参照される。
LC_CTYPE	データ型変換に使用する文字セット	Client-Library アプリケーションまたは Open Server アプリケーション	アプリケーションが *type* パラメータに CS_LC_CTYPE、*buffer* パラメータに NULL を指定して cs_locale を呼び出す場合に参照される。
LC_MESSAGE	メッセージに使用する言語と文字セット	Client-Library アプリケーションまたは Open Server アプリケーション	アプリケーションが *type* パラメータに CS_LC_MESSAGE、*buffer* パラメータに NULL を指定して cs_locale を呼び出す場合に参照される。
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。	Client-Library アプリケーションまたは Open Server アプリケーション	アプリケーションが *type* パラメータに CS_LC_TIME、*buffer* パラメータに NULL を指定して cs_locale を呼び出す場合に参照される。

環境変数	使用目的	使用箇所	参照箇所
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/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。	Client-Library アプリケーションまたは Open Server アプリケーション	アプリケーションが cs_ctx_alloc または cs_ctx_global を呼び出したときに LC_ALL 環境変数が定義されていない場合、Client-Library は LANG 環境変数を調べる。 アプリケーションが cs_locale を呼び出したときに cs_locale の *buffer* パラメータが NULL であり、*type* パラメータに対応する LC 環境変数が定義されていない場合、Client-Library は LANG 環境変数を調べる。
		Embedded SQL プリコンパイラ	アプリケーションのプリコンパイル時に LC_ALL 環境変数が定義されていない場合に参照される。
		プリコンパイルされた Embedded SQL アプリケーション	アプリケーション実行時に LC_ALL 環境変数が定義されていない場合に参照される。

環境変数を使用しないプラットフォーム

この項では、環境変数を使用しないプラットフォームについて説明します。

デスクトップ用語

一部のプラットフォームでは、「環境変数」の代わりに「環境値」という用語が使用されます。これらの用語は同義語です。

CS_LOCALE 構造体

CS_LOCALE 構造体は、言語、文字セット、照合順、日時フォーマットを含むローカライゼーション情報の完全な集合を保管します。

Open Client/Server アプリケーションでは、コンテキスト、接続、スレッド、データ要素、またはルーチンにカスタム・ローカライゼーション値を定義するには CS_LOCALE 構造体を使用する必要があります。

❖ **CS_LOCALE 構造体を使用するには、次の手順に従います。**

1. cs_loc_alloc を呼び出して、CS_LOCALE 構造体を割り付けます。
2. cs_locale を呼び出し、適切なローカライゼーション値を使用して CS_LOCALE 構造体をロードします。「cs_locale ルーチン」(12 ページ) を参照してください。

3 必要に応じて cs_dt_info(CS_SET,CS_DT_CONVFMT) を呼び出して、CS_LOCALE 構造体の日付変換フォーマットを変更します。『Open Client/Server Common Libraries リファレンス・マニュアル』を参照してください。

4 ロードされた CS_LOCALE 構造体を使用して、コンテキスト、接続、スレッド、データ要素、またはルーチンをカスタマイズします。

- コンテキストをカスタマイズするには、cs_config を呼び出します。
- 接続をカスタマイズするには、ct_con_props を呼び出します。
- スレッドをカスタマイズするには、srv_thread_props を呼び出します。
- データ要素にカスタム値を定義するには、CS_DATAFMT 構造体内で CS_LOCALE へのポインタを指定します。

5 ルーチンにカスタム値を定義するには、CS_LOCALE 構造体へのポインタをそのルーチンに渡します。

cs_locale ルーチン

Open Client/Server アプリケーションは cs_locale ルーチンを呼び出し、カスタム・ローカライゼーション情報を使用して CS_LOCALE 構造体をロードします。

cs_locale は次のようにして宣言します。

```
CS_RETCODE cs_locale(context, action, locale, type,
buffer, buflen, outlen)

 CS_CONTEXT *context;
 CS_INT action;
 CS_LOCALE *locale;
 CS_INT type;
 CS_CHAR *buffer;
 CS_INT buflen;
 CS_INT *outlen;
```

cs_locale は、呼び出されると、次のタスクを行います。

1 どのロケール名を使用するかを決定します。

cs_locale の *buffer* パラメータが指定された場合は、そのパラメータがロケール名になります。

cs_locale の *buffer* パラメータが NULL の場合、cs_locale はその *type* パラメータに対応する環境変数を調べて、この環境変数の値をロケール名として使用します。適切な環境変数の値が *locales.dat* ファイル内のエントリと対応していることを確認してください。

type パラメータに対応する環境変数が設定されていない場合、cs_locale は “default” のロケール名を使用します。

2　*locales.dat* ファイルのロケール名を検索して、関連付けられている言語、文字セット、照合順を決定します。cs_locale が一致するエントリを検出できない場合は、CS_FAIL を返します。

3　cs_locale の *type* パラメータで指定された情報を CS_LOCALE 構造体にロードします。たとえば *type* パラメータが CS_LC_CTYPE である場合、cs_locale は文字セット情報をロードします。

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

例：*cs_locale* を呼び出して CS_LOCALE 構造体をロードする場合

次のエントリが記述されている *locales.dat* ファイルのあるマシン上でアプリケーションが稼働していると仮定します。

```
locale = korean, korean, eucksc, korsrt
locale = C.korean, us_english, eucksc, ussrt
locale = default, us_english, iso_1, ussrt
```

ここで、エントリのフォーマットは次のとおりです。

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

環境変数 LC_MESSAGE の値が “korean” であり、環境変数 LC_TIME は定義されていないと仮定します。この環境ではアプリケーションが 2 回 cs_locale を呼び出して、次のカスタム値を使用して CS_LOCALE 構造体をロードする必要があります。

- Client-Library とサーバ・メッセージの言語には “korean”、文字セットには “eucksc”
- 日時の値の変換に使用する言語には “us_english”、文字セットには “eucksc”

この 2 回の cs_locale 呼び出しを次に示します。

```
/*
 ** You should not specify a locale name, because
 ** cs_locale will use the value of the LC_MESSAGE
 ** environment variable as the locale name.
 */
 cs_locale(ctx, CS_SET, mylocale, CS_LC_MESSAGE,
     NULL, CS_UNUSED, NULL);

 /* Do need to specify a locale name, because
 ** there's no LC_TIME environment variable set.
 */
 cs_locale(ctx, CS_SET, mylocale, CS_LC_TIME,
     "C.korean", CS_NULLTERM, NULL);
```

CS_LOCALE 構造体をロードしたあとで、アプリケーションは次のタスクを行うことができます。

- cs_config を呼び出して、コンテキスト構造体にカスタム・ローカライゼーション値をコピーできます。
- ct_con_props を呼び出して、接続構造体にカスタム・ローカライゼーション値をコピーできます。
- srv_thread_props を呼び出して、スレッド構造体にカスタム・ローカライゼーション値をコピーできます。
- カスタム・ローカライゼーション値を受け取るルーチンへのパラメータとして CS_LOCALE 構造体を指定できます (cs_strcmp、cs_time)。
- 変換元または変換先のプログラム変数を記述する CS_DATAFMT 構造体に CS_LOCALE を含めることができます (cs_convert、ct_bind)。

第 3 章

国際化された Open Client/Server アプリケーションの開発

この章では、国際化された Open Client/Server アプリケーションの開発方法について説明します。

この章の内容は、次のとおりです。

この章には、Open Client/Server アプリケーションを開発するために必要な情報がすべて示されているわけではありません。この章で示されていない情報については、次のマニュアルとアプリケーションを参照してください。

- 『Open Client/Server Common Libraries リファレンス・マニュアル』
- 『Open Client Client-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』
- Open Client/Server 製品に付属している国際化サンプル・アプリケーション (Open Client 用の *i18n.c* と Open Server 用の *intlchar.c*)

  詳細については、使用してい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』を参照してください。

国際化 Client-Library アプリケーションの開発

アプリケーションをどのようにローカライズするか、つまり、どの環境でどの言語、文字セット、文化的慣例を使用するかを決定してから、国際化された Client-Library アプリケーションを開発する必要があります。

Client-Library アプリケーションは初期ローカライゼーション値またはカスタム・ローカライゼーション値、あるいはその両方を使用できます。

大部分の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。

初期ローカライゼーション値がどのように決定されるか、およびアプリケーションがこれらの値を使用できるかどうかを判断する方法については、「使用するローカライゼーション値の決定」(5 ページ) を参照してください。

初期値を使用する Client-Library アプリケーション

アプリケーションが初期ローカライゼーション値を使用する場合は、アプリケーションを国際化するために特別なコードを指定しないでください。

アプリケーションを分散する場合は、システム管理者が環境変数の設定方法を理解している必要があります。「初期ローカライゼーション値を使用するアプリケーションの設定」(7 ページ) を参照してください。

カスタム値を使用する Client-Library アプリケーション

Client-Library アプリケーションは、コンテキスト・レベル、接続レベル、データ要素レベルでカスタム・ローカライゼーション値を使用できます。

Open Client/Server アプリケーションは、次の手順に従ってカスタム・ローカライゼーション値を設定します。

- cs_locale を呼び出し、特定のローカライゼーション値を使用して CS_LOCALE 構造体をロードします。
- ロードされた CS_LOCALE 構造体を使用して、コンテキスト、接続、またはデータ要素をカスタマイズします。

表 3-1 は、アプリケーションでカスタム・ローカライゼーション値の使用方法を決定するときに参考にしてください。

表 3-1: *Client-Library* アプリケーションでのカスタム・ローカライゼーション値の使用

条件	作業	参照箇所
アプリケーションが 1 組のカスタム・ローカライゼーション値だけを必要とする (ただし、どのような理由があっても初期ローカライゼーション値を使用できない) 場合。	コンテキスト・レベルでカスタマイズする。 同じ CS_LOCALE 構造体を使用して、複数のコンテキストをカスタマイズできる。	「コンテキスト・レベルでのカスタマイズ」(17 ページ)
アプリケーション内のさまざまなコンテキストに対し、異なるローカライゼーション値が必要な場合。	それぞれのコンテキストをカスタマイズする。 別個の CS_LOCALE 構造体を使用して、さまざまなコンテキストをカスタマイズする。	「コンテキスト・レベルでのカスタマイズ」(17 ページ)
特定の接続が親コンテキストのローカライゼーション値とは異なるローカライゼーション値を使用する必要がある場合。	それらの接続をカスタマイズする。	「接続レベルでのカスタマイズ」(18 ページ)
バインド変数、変換先変数、または特定のルーチンがカスタム・ローカライゼーション値を使用する必要がある場合。	変数またはルーチンをカスタマイズする。	「データ要素レベルでのカスタマイズ」(20 ページ)

コンテキスト・レベルでのカスタマイズ

コンテキストの初期ローカライゼーション値が適切でない場合は、コンテキスト・レベルでカスタム・ローカライゼーション値をインストールする必要があります。

たとえば、同じアプリケーション内のさまざまなコンテキストに対し、異なるローカライゼーション値が必要な場合、コンテキスト・レベルでカスタム・ローカライゼーション値をインストールする必要があります。これは、すべてのコンテキストが正しい初期値を使用して作成されるとは限らないからです。

コンテキストが初期ローカライゼーション値をどのように受け取るかについては、「初期ローカライゼーション値の使用」(6 ページ) を参照してください。

例

ある Client-Library アプリケーションが韓国語でメッセージを生成する必要があり、その一方で他のアプリケーションを受け入れるために LC_ALL 環境変数が us_english に設定される必要がある環境で稼働していると仮定します。この場合、コンテキストが使用する初期の us_english ローカライゼーション値は適切ではないので、アプリケーションはコンテキスト・レベルで韓国語のローカライゼーション値を指定する必要があります。

コンテキスト・レベルでのカスタム・ローカライゼーション値の定義

表 3-2 は、コンテキスト・レベルでカスタム・ローカライゼーション値を定義する方法について説明しています。

表 3-2: コンテキスト・レベルでのカスタム値のインストール

手順	アプリケーションの手順	目的	詳細
1	cs_loc_alloc を呼び出す。	CS_LOCALE 構造体を割り付ける。	この呼び出しによって、親コンテキストの現在のローカライゼーション情報は CS_LOCALE 構造体にコピーされる。
2	cs_locale を呼び出す。	カスタム・ローカライゼーション値を使用して CS_LOCALE 構造体を上書きする。	「cs_locale ルーチン」(12 ページ) を参照。 Open Server アプリケーションは、*type* を CS_LC_ALL に設定して cs_locale を呼び出す必要がある。これによって、Server-Library が内部的に一貫したローカライゼーション値を使用して CS_LOCALE 構造体をロードすることが保証される。
3	オプションで cs_dt_info を呼び出す。	CS_LOCALE 構造体内の日時変換フォーマットを変更する。	『Open Client/Server Common Libraries リファレンス・マニュアル』 を参照してください。
4	*property* パラメータに CS_LOC_PROP を指定して cs_config を呼び出す。	コンテキストをカスタマイズする。	
5	オプションで cs_loc_drop を呼び出す。	CS_LOCALE 構造体の割り付けを解除する。	アプリケーションは、CS_LOCALE 構造体の割り付けを解除する前に構造体を再使用できる。 必要に応じてアプリケーションは cs_locale を呼び出して、構造体内のローカライゼーション値を変更してから、その構造体を再使用できる。

接続レベルでのカスタマイズ

接続は、その親コンテキストからデフォルトのローカライゼーション値を継承します。接続のデフォルト・ローカライゼーション値が適切でない場合は、接続レベルでカスタム・ローカライゼーション値をインストールする必要があります。

例

スペインにあるサーバに接続する us_english/iso1 アプリケーションが roman8 の文字データを処理、ソートする必要があると仮定します。この場合、接続がその親コンテキストから継承した us_english/iso1 のローカライゼーション値では適切ではないので、アプリケーションは接続レベルで roman8 のローカライゼーション値をインストールする必要があります。

接続レベルでのカスタム・ローカライゼーション値の定義

表 3-3 は、接続レベルでカスタム・ローカライゼーション値を定義する方法について説明しています。

表 *3-3:* 接続レベルでのカスタム値のインストール

手順	アプリケーションの手順	目的	詳細
1	cs_loc_alloc を呼び出す。	CS_LOCALE 構造体を割り付ける。	この呼び出しによって、親コンテキストの現在のローカライゼーション情報は CS_LOCALE 構造体にコピーされる。
2	cs_locale を呼び出す。	カスタム・ローカライゼーション値を使用して CS_LOCALE 構造体を上書きする。	「cs_locale ルーチン」(12 ページ) を参照。
3	オプションで cs_dt_info を呼び出す。	CS_LOCALE 構造体内の日時変換フォーマットを変更する。	『Open Client/Server Common Libraries リファレンス・マニュアル』を参照。
4	*property* パラメータに CS_LOC_PROP を指定して ct_con_props を呼び出す。	接続をカスタマイズする。	CS_LOC_PROP がログイン・プロパティであることに注意すること。接続がオープンされたあとでは、アプリケーションはその値を変更できない。 接続がオープンされたあとにアプリケーションがその接続の言語または文字を変更する要求をサーバに送信した場合、その変更は CS_LOC_PROP の値には反映されない。アプリケーションが ct_con_props を呼び出して、CS_LOC_PROP の値を検索する場合、検索されたロケール構造体にはその接続の現在のローカライゼーション値は含まれない。
5	オプションで cs_loc_drop を呼び出す。	CS_LOCALE 構造体の割り付けを解除する。	アプリケーションは、CS_LOCALE 構造体の割り付けを解除する前に構造体を再使用できる。 構造体のローカライゼーション値を再利用する前に、必要に応じてアプリケーションは cs_locale を呼び出して、この値を変更できる。

クライアント・アプリケーションが ct_connect を呼び出して接続をオープンすると、サーバは要求されたローカライゼーションをサポートできるかどうかを調べます。サポートできる場合、サーバは接続をそのまま受け入れます。サポートできない場合、サーバは強制的にその接続が代替言語か代替文字セット、またはその両方を使用するよう命令します。ここで、変更された接続をクライアントが受け入れる場合もあれば、拒否する場合もあります。

データ要素レベルでのカスタマイズ

データ要素のローカライゼーション値を使用して、次のものをカスタマイズできます。

- バインド変数 (ct_bind)

  カスタム・ローカライゼーション値が指定されない場合、バンド変数は対応する接続からのローカライゼーション値を使用します。

- 変換先変数 (cs_convert)

  カスタム・ローカライゼーション値が指定されない場合、変換先変数は cs_convert の *context* パラメータのローカライゼーション値を使用します。

- cs_time と cs_strcmp の動作

  カスタム・ローカライゼーション値が指定されない場合、これらのルーチンは *context* パラメータに対応するローカライゼーション値を使用します。

デフォルト値が適切でない場合は、データ要素レベルでカスタム・ローカライゼーション値を設定する必要があります。

例

us_english での接続を伴うアプリケーションが、レポートを作成するために us_english のデータベースから本のタイトルと発行日を選択すると仮定します。このレポートはパリに送信されるので、発行日はフランス語の標準フォーマットになっている必要があります。

日付カラムのバインド変数としてはその接続の us_english フォーマットは適切ではないので、アプリケーションはフランス語の日時フォーマットを使用するようにバインド変数を設定する必要があります。

アプリケーションは、フランス語の日時フォーマットを使用するように、日付カラムのバインド変数を次のようにして設定できます。

- アプリケーションは、フランス語の日時フォーマットを使用して CS_LOCALE 構造体をロードします。
- アプリケーションは ct_bind を呼び出して、文字変数に日付カラムをバインドします。ct_bind 呼び出しでは、バインド変数を記述する CS_DATAFMT 構造体はフランス語の日時フォーマットを使用して CS_LOCALE 構造体を参照します。

アプリケーションが ct_fetch を呼び出すと、日付カラムの日時の値は、フランス語の日付と月の名前を含む文字列に自動的に変換され、バインドされた変数に自動的にコピーされます。

データ要素レベルでのカスタム・ローカライゼーション値の定義

表 3-4 は、データ要素レベルでカスタム・ローカライゼーション値を定義する方法について説明しています。

表 *3-4:* データ要素レベルでのカスタム値のインストール

手順	アプリケーションの手順	目的	詳細
1	cs_loc_alloc を呼び出す。	CS_LOCALE 構造体を割り付ける。	この呼び出しによって、親コンテキストの現在のローカライゼーション情報は CS_LOCALE 構造体にコピーされる。
2	cs_locale を呼び出す。	カスタム・ローカライゼーション値を使用して CS_LOCALE 構造体を上書きする。	「cs_locale ルーチン」(12 ページ) を参照。
3	オプションで cs_dt_info を呼び出す。	CS_LOCALE 構造体内の日時変換フォーマットを変更する。	『Open Client/Server Common Libraries リファレンス・マニュアル』を参照。
4	CS_LOCALE 構造体を使用する。	バインド変数、変換先変数、またはルーチンをカスタマイズする。	• ct_bind の *datafmt* パラメータを CS_LOCALE 構造体にして、変換先変数をカスタマイズする。 • cs_convert の *destfmt* パラメータを CS_LOCALE 構造体にして、変換先変数をカスタマイズする。 • ルーチンへのパラメータとして CS_LOCALE 構造体を指定して、cs_strcmp または cs_time の動作をカスタマイズする。
5	オプションで cs_loc_drop を呼び出す。	CS_LOCALE 構造体の割り付けを解除する。	CS_DATAFMT 構造体が CS_LOCALE 構造体を参照しているうちは、アプリケーションは CS_LOCALE 構造体の割り付けを解除してはならない。

Client-Library のローカライゼーション値の優先度

Client-Library は、次の優先度でローカライゼーション値を使用します。

1　データ要素レベルで定義された値

2　接続レベルで定義された値

3　コンテキスト・レベルで定義された値

Client-Library のローカライゼーション・プロパティ

表 3-5 は、ローカライゼーションに関係のある Client-Library プロパティを示しています。

表 3-5: ローカライゼーションに関係のある *Client-Library* プロパティ

プロパティ	説明	適用対象	参照箇所
CS_LOC_PROP	ローカライゼーション情報を定義する CS_LOCALE 構造体	コンテキスト、接続	『Open Client Client-Library/C リファレンス・マニュアル』
CS_CHARSETCNV	サーバが文字セットの変換を実行するかどうかを決定する	接続	『Open Client Client-Library/C リファレンス・マニュアル』
CS_NOCHARSETCNV	サーバが文字セットの変換を実行する必要があるかどうかを決定する	接続	『Open Client Client-Library/C リファレンス・マニュアル』

国際化された Open Server アプリケーションの開発

国際化された Open Server アプリケーションを開発する場合は、次の点を考慮してください。

- アプリケーション自体をどのようにローカライズするか
- アプリケーションがローカライズされたクライアントをどのようにサポートするか
- 言語と文字セットを変更するクライアント要求にアプリケーションがどのように応答するか
- Server-Library のローカライゼーション・プロパティがどのような値である必要があるか

アプリケーションのローカライズ

Open Server アプリケーションのローカライゼーション値は、エラー・メッセージが生成される言語、すべてのデータ操作に使用される文字セット、照合順を決定します。

注意 SRV_S_USESRVLANG プロパティと SRV_T_USESRVLANG プロパティは、エラー・メッセージを生成するときにサーバの言語を上書きするのに使用できます。

Open Server アプリケーションは初期ローカライゼーション値やカスタム・ローカライゼーション値、またはその両方を使用することもできます。

大部分の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。

初期ローカライゼーション値は、アプリケーションがそのコンテキスト構造体を割り付けるときに決定されます。アプリケーションが初期ローカライゼーション値を使用できるかどうかを決定する方法については、「使用するローカライゼーション値の決定」(5 ページ) を参照してください。

初期値を使用する Open Server アプリケーション

アプリケーションが初期ローカライゼーション値を使用する場合は、アプリケーションを国際化するために特別なコードを指定しないでください。

アプリケーションを分散する場合は、システム管理者が環境変数の設定方法を理解している必要があります。「初期ローカライゼーション値を使用するアプリケーションの設定」(7 ページ) を参照してください。

カスタム値を使用する Open Server アプリケーション

アプリケーションが初期ローカライゼーション値を使用できない場合は、アプリケーションワイドなコンテキスト構造体内にカスタム・ローカライゼーション情報をインストールしてから、srv_version を呼び出す必要があります。この実行方法の詳細については、表 3-2 (18 ページ) を参照してください。

ローカライズされたクライアントのサポート

Open Server はローカライズされたクライアントに対して自動的にいくつかのサポートを提供しますが、アプリケーションがサポートを追加する必要がある場合もあります。

ローカライズされたクライアントに対する自動サポート

Open Server はローカライズされたクライアントのサポートに関係のあるいくつかのタスクを自動的に処理します。これらのタスクには次のようなものがあります。

- 要求があれば受信データと送信データの両方の文字セット変換を実行します。
- (クライアントのスレッド構造体の SRV_T_USESRVLANG プロパティが CS_FALSE に設定されている場合) クライアントの言語と文字セットで Open Server のエラー・メッセージを表示します。
- クライアント要求に応えてローカライゼーション情報をクライアントに提供します。「ローカライゼーション情報に関する要求への自動応答」(24 ページ) を参照してください。

Open Server アプリケーションの中には、ローカライズされたクライアントをサポートするための追加手順の必要がないので、ローカライズされたクライアントに対するこの自動サポートで十分なものもあります。しかし、ローカライズされたクライアントに対して追加のサポートが必要な Open Server アプリケーションもあります。

ローカライゼーション情報に関する要求への自動応答

Open Server アプリケーションにログインした後で、クライアントは次の情報を要求できます。

- サーバの文字セットの名前
- サーバの照合順 (ソート順) の名前
- クライアント文字セットのための文字セットの定義
- クライアントの照合順のためのソート順定義

クライアントは、RPC (リモート・プロシージャ・コール) コマンドを使用して、sp_serverinfo システム・レジスタード・プロシージャによってこれらの情報を要求します。

Open Server からの応答として、sp_serverinfo システム・レジスタード・プロシージャを介して要求された情報が自動的に送り返されます。Open Server アプリケーションはこの時点で何も実行する必要はなく、実際には要求があったことも認識していません。

ローカライズされたクライアントのための追加サポート

次の場合、ローカライズされたクライアントをサポートするには Open Server アプリケーションがさらに追加の手順を実行する必要があります。

- Open Server アプリケーションが CS-Library のエラー・メッセージをクライアントに渡す場合

  この場合、Open Server アプリケーションは、CS-Library がクライアントの言語と Open Server アプリケーションの文字セットでメッセージを生成していることを確認する必要があります。この実行方法の詳細については、「クライアントに対する CS-Library メッセージのローカライズ」(25 ページ) を参照してください。

- Open Server アプリケーションがゲートウェイとして機能する場合

  この場合、Open Server アプリケーションは、リモート・サーバへの接続がクライアントの言語と Open Server アプリケーションの文字セットを使用していることを確認する必要があります。この実行方法の詳細については、「Open Server ゲートウェイに対してローカライズされた接続の作成」(27 ページ) を参照してください。

- クライアント・アプリケーションが言語または文字セットの変更を要求した場合

  この場合、Open Server アプリケーションは、クライアント・スレッドの言語または文字セットを変更する必要があります。この実行方法の詳細については、「言語と文字セットの変更要求への応答」(27 ページ) を参照してください。

クライアントに対する CS-Library メッセージのローカライズ

Open Server アプリケーションがアプリケーション自体のコンテキスト構造体をパラメータとして使用して CS-Libary ルーチンを呼び出すと、呼び出し結果として CS-Library が生成するエラー・メッセージでは、Open Server アプリケーションの言語と文字セットが使用されます。

たとえば、cs_convert 呼び出しのコンテキスト・パラメータが us_english/iso_1 を示している場合、cs_convert 呼び出しが失敗すると、CS-Library は us_english/iso_1 のメッセージを生成します。

注意　CS-Library ルーチンのパラメータとして CS_LOCALE 構造体が指定されると、この構造体の中のローカライゼーション値によってコンテキスト・パラメータの中のローカライゼーション値が上書きされます。

Open Server アプリケーションの言語と文字セットで CS-Library メッセージを取得できるのは、Open Server アプリケーションが CS-Library メッセージのログを取るか、または CS-Library メッセージを保持する場合のみです。

しかし、Open Server アプリケーションが CS-Library のエラー・メッセージをクライアントに送り返す場合、CS-Library はクライアントの言語と Open Server アプリケーションの文字セットでメッセージを生成します。

メッセージは、クライアントが理解できるクライアントの言語で生成される必要があります。

次の 2 つの理由から、メッセージは Open Server アプリケーションの文字セットでなければなりません。

- Open Server アプリケーションは一般にすべてのメッセージをログ・ファイルに記録します。したがって、ログに記録されたすべてのメッセージが同一の文字セットを使用することが重要です。
- Open Server は、メッセージを含む送信データの文字セット変換を自動的に実行します。Open Server の文字セットでメッセージを生成することによって、そのメッセージがクライアントの文字セットに正しく変換されることが保証されます。

アプリケーションは、クライアント・スレッドごとに正しくローカライズされた CS_CONTEXT 構造体を設定することによって、メッセージを正しい言語および文字セットで確実に生成します。また、クライアントの代わりに CS-Library ルーチンを呼び出すときに、これらの CS_CONTEXT 構造体を使用できます。

クライアント・スレッドに対する CS_CONTEXT 構造体のローカライズ

表 3-6 は、クライアント・スレッドに対して CS_CONTEXT 構造体をローカライズする方法を示しています。

表 *3-6:* クライアント・スレッドに対する *CS_CONTEXT* 構造体のローカライズ

手順	アプリケーションの手順	目的	詳細
1	cs_ctx_alloc を呼び出す。	クライアント・スレッドに対して CS_CONTEXT 構造体を割り付ける。	コンテキスト構造体は、初期ローカライゼーション値を使用して割り付けられる。
2	cs_loc_alloc を呼び出す。	新しい CS_LOCALE 構造体を割り付ける。	この呼び出しによって、親コンテキストの現在のローカライゼーション情報は新しい CS_LOCALE 構造体にコピーされる。
3	*property* を SRV_T_LOCALE に設定して srv_thread_props(GET) を呼び出す。	クライアント・スレッドの既存のローカライゼーション値を新しい CS_LOCALE 構造体にコピーする。	
4	*type* パラメータに CS_SYB_CHARSET を指定して cs_locale を呼び出す。	新しい CS_LOCALE 構造体内のクライアント・スレッドの文字セット情報を Open Server アプリケーションの文字セット情報と置き換える。	
5	*property* パラメータに CS_LOC_PROP を指定して cs_config を呼び出す。	コンテキスト構造体をカスタマイズする。	この呼び出しによって、ローカライゼーション情報は CS_LOCALE 構造体から CS_CONTEXT 構造体にコピーされる。
6	オプションで cs_loc_drop を呼び出す。	CS_LOCALE 構造体の割り付けを解除する。	アプリケーションは、CS_LOCALE 構造体の割り付けを解除する前に構造体を再使用できる。 必要に応じてアプリケーションは cs_locale を呼び出して、構造体内のローカライゼーション値を変更してから、その構造体を再使用できる。

Open Server ゲートウェイに対してローカライズされた接続の作成

Open Server アプリケーションがゲートウェイとして機能している場合は、リモート・サーバへの接続がクライアントの言語と Open Server の文字セットを使用することを保証する必要があります。

注意　Open Server の文字セットは、リモート・サーバの文字セットと同じである必要はありませんが、リモート・サーバがそのサーバ自体の文字セットに変換可能なものである必要があります。

Adaptive Server Enterprise は、2 つの西欧文字セット間での変換や 2 つの日本語文字セット間での変換はできますが、西欧文字セットから日本語文字セットへの変換はできません (また、その逆の日本語文字セットから西欧文字セットへの変換もできません)。たとえば Adaptive Server Enterprise は、ともに西欧言語グループの文字セットである ISO 8859-1 と CP850 との間では変換できますが、西欧言語グループの文字セットである ISO 8859-1 と、東欧言語グループの文字セットである CP 1250 との間での変換は実行できません。ただし、Adaptive Server Enterprise が異なる言語グループの文字セット間での変換を行う場合は、ASCII 以外の文字は失われる可能性があります。

アプリケーションが上記のタスクを実行する場合の最も簡単な方法は、各クライアント・スレッドに対して適切にローカライズされた CS_CONTEXT 構造体を設定してから、ローカライズされたコンテキスト内にクライアント・スレッドのリモート接続を割り付ける方法です。

「クライアント・スレッドに対する CS_CONTEXT 構造体のローカライズ」(26 ページ) を参照してください。

接続の割り付け方法の詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

言語と文字セットの変更要求への応答

クライアントが Open Server アプリケーションに接続するときに、Open Server は、クライアントの言語と文字セットを反映する CS_LOCALE 構造体を自動的に作成します (クライアントの照合順は CS_LOCALE 構造体に含まれていません。照合順の情報はログイン時にサーバには転送されません)。

たとえば french/cp850 を使用するクライアントが us_english/iso_1 を使用する Open Server アプリケーションにログインする場合、Open Server アプリケーションは french/cp850 の CS_LOCALE 構造体を作成します。Open Server アプリケーションはこの CS_LOCALE 構造体を使用して、クライアント・スレッドの文字セット変換ルーチンを設定します。

注意　この CS_LOCALE 構造体内の情報は、Open Server プログラマが使用できます。Open Server プログラマは srv_thread_props を呼び出して、新しく割り付けられた CS_LOCALE 構造体に情報をコピーできます。

ログイン後にクライアントが言語または文字セットを変更する要求を送信した場合、Open Server アプリケーションは要求された変更をクライアント・スレッドの CS_LOCALE 構造体内で行う必要があります。

クライアントは、次の 2 つの方法のいずれかで言語または文字セットの変更を要求できます。

- (ct_command を使用して送信される) 言語に基づいたオプション・コマンドを使用する。このタイプのコマンドは SRV_LANGUAGE イベントをトリガします。その結果、Open Server アプリケーションは SRV_LANGUAGE イベント・ハンドラの内部で要求を処理します。
- (ct_options を使用して送信される) オプション・コマンドを使用する。このタイプのコマンドは SRV_OPTION イベントをトリガします。その結果、Open Server アプリケーションは SRV_OPTION イベント・ハンドラの内部で要求を処理します。

表 3-7 は、クライアント・スレッドの言語または文字セットを変更する方法を示します。

表 *3-7:* クライアント・スレッドの言語または文字セットの変更

手順	アプリケーションの手順	目的	詳細
1	cs_loc_alloc を呼び出す。	CS_LOCALE 構造体を割り付ける。	この呼び出しは、Open Server アプリケーション・コンテキストの現在のローカライゼーション情報を新しい CS_LOCALE 構造体にコピーする。
2	*property* を SRV_T_LOCALE に設定して srv_thread_props (GET) を呼び出す。	クライアント・スレッドの既存のローカライゼーション値を新しい CS_LOCALE 構造体にコピーする。	
3	cs_locale を呼び出す。	要求された言語または文字セットを使用して、CS_LOCALE 構造体を上書きする。	「cs_locale ルーチン」(12 ページ) を参照。
4	*property* を SRV_T_LOCALE に設定して srv_thread_props (SET) を呼び出す。	新しい言語または文字セットを使用して、クライアント・スレッドを設定する。	
5	オプションで cs_loc_drop を呼び出す。	CS_LOCALE 構造体の割り付けを解除する。	アプリケーションは、CS_LOCALE 構造体の割り付けを解除する前に構造体を再使用できる。 必要に応じてアプリケーションは cs_locale を呼び出して、構造体内のローカライゼーション値を変更してから、その構造体を再使用できる。

注意　Open Server および SDK は、Adaptive Server Enterprise と同じ文字セットをサポートしています。

Server-Library のローカライゼーション・プロパティ

表 3-8 は、ローカライゼーションに関係のある Server-Library プロパティを示しています。

表 *3-8:* ローカライゼーションに関係のある *Server-Library* プロパティ

プロパティ	説明	適用対象	参照箇所
SRV_S_USESRVLANG	サーバの言語でメッセージを生成するかどうか。	アプリケーションワイドなコンテキスト	『Open Server Server-Library/C リファレンス・マニュアル』
SRV_T_USESRVLANG	サーバの言語でメッセージを生成するかどうか。	スレッド	『Open Server Server-Library/C リファレンス・マニュアル』

これらのプロパティは、Open Server がエラー・メッセージを Open Server アプリケーションの言語で生成するかクライアントの言語で生成するかを指定します。

SRV_S_USESRVLANG はサーバワイドなプロパティであり、srv_props を使用して設定されます。この値は、SRV_T_USESRVLANG のデフォルト値として使用されます。

SRV_T_USESRVLANG はスレッドのプロパティであり、srv_thread_props を使用して設定されます。新しいスレッド構造体が割り当てられたとき、SRV_T_USESRVLANG は SRV_S_USESRVLANG からデフォルト値を抽出します。

- SRV_T_USESRVLANG が CS_TRUE である場合、Open Server はサーバの言語でスレッドについてのエラー・メッセージを生成します。
- SRV_T_USESRVLANG が CS_FALSE である場合、Open Server はクライアントの言語でスレッドについてのエラー・メッセージを生成します。

国際化された DB-Library アプリケーションの開発

新しいクライアント・アプリケーションを開発する場合、プログラマは DB-Library の代わりに Client-Library を使用してください。この項の内容は、既存の DB-Library アプリケーションを使用しているサイトを対象にしています。

Client-Library とは異なり、DB-Library では環境変数を調べて初期ローカライゼーション値を決定することはありません。その代わり DB-Library では、初期ローカライゼーション値はプラットフォームごとにあらかじめ定義されています。

アプリケーションは接続をオープンするのに使用されるログイン・レコード内の言語名と文字セット名を次のように変更して、特定の接続に対するこれらの初期値を変更できます。

- 言語名を変更するには、DBSETLNATLANG (*login,language_name*) を呼び出します。
- 文字セット名を変更するには、DBSETLCHARSET (*login,charset_name*) を呼び出します。アプリケーションは DBSETLCHARSET (*login,*NULL) を呼び出して、サーバが文字セット変換を実行しないように指定できます。

アプリケーションは、それぞれのサーバ接続に対して別々の言語と文字セットを使用できます。

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

Embedded SQL での国際化

Embedded SQL アプリケーション・プログラマは、次のアプリケーションをローカライズできます。

- Embedded SQL プリコンパイラ
- プリコンパイルされた Embedded SQL アプリケーション

プリコンパイラのローカライズ

プリコンパイラ・ユーザはプリコンパイラをデフォルトのローカライゼーション値を使用して実行することもできますし、カスタム・ローカライゼーション値を使用して実行することもできます。

デフォルト値の決定方法

コマンド・ライン・オプションを指定しない場合、プリコンパイラのローカライゼーション値はプリコンパイラの実行時に次のようにして決定されます。

- LC_ALL 環境変数を設定した場合、アプリケーションはその値を使用してローカライズし、LC_ALL 環境変数の値がロケール・ファイル内のエントリと一致するかどうかを調べてどの言語と文字セットを使用するかを決定します。
- LC_ALL 環境変数を設定しないで LANG 環境変数を設定した場合、アプリケーションはその値を使用してローカライズし、LANG 環境変数の値がロケール・ファイル内のエントリと一致するかどうかを調べてどの言語と文字セットを使用するかを決定します。
- 上記のどちらの環境変数も設定しない場合、アプリケーションはロケール・ファイル内の “default” エントリを使用します。

カスタム・ローカライゼーション値の指定

プリコンパイラ・ユーザはコマンド・ライン・オプションを使用して、次のものに対してカスタム・ローカライゼーション値を指定できます。

- ソース・ファイルの文字セット

  プリコンパイルされるソース・ファイルの文字セットを指定するには、次のコマンド・ライン・オプションを使用してください。

  ```
  -J  locale_for_charset
  ```

  ここで *locale_for_charset* には、ロケール・ファイル内にエントリが指定されているロケール名が入ります。

  -J を指定しない場合、プリコンパイラはソース・ファイルがプリコンパイラのデフォルト文字セットを使用していると解釈します。

- プリコンパイラのメッセージ

  プリコンパイラがメッセージに使用する言語と文字セットを指定するには、次のコマンド・ライン・オプションを使用してください。

  ```
  -Z locale_for_messages
  ```

  ここで *locale_for_messages* には、ロケール・ファイル内にエントリが指定されているロケール名が入ります。

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

Embedded SQL アプリケーションのローカライズ

Embedded SQL アプリケーションのローカライゼーション値は、アプリケーションの実行時に次のように決定されます。

- LC_ALL 環境変数を設定した場合、アプリケーションはその値を使用してローカライズし、LC_ALL 環境変数の値がロケール・ファイル内のエントリと一致するかどうかを調べてどの言語と文字セットを使用するかを決定します。
- LC_ALL 環境変数を設定しないで LANG 環境変数を設定した場合、アプリケーションはその値を使用してローカライズし、LANG 環境変数の値がロケール・ファイル内のエントリと一致するかどうかを調べてどの言語と文字セットを使用するかを決定します。
- 上記のどちらの環境変数も設定しない場合、アプリケーションはロケール・ファイル内の “default” エントリを使用します。

一般的な Embedded SQL アプリケーションは、LC_ALL 環境変数を設定してローカライズします。

スタンドアロン・ユーティリティのローカライズ

スタンドアロン・ユーティリティには isql、bcp、defncopy などがあります。Client-Library で構築されたユーティリティと DB-Library で構築されたユーティリティは、異なる方法でローカライズします。

Client-Library で構築されたユーティリティは、環境変数を調べてデフォルトのローカライゼーション値を決定します。「使用するローカライゼーション値の決定」(5 ページ) と「初期ローカライゼーション値の使用」(6 ページ) を参照してください。

DB-Library を使用して構築されたユーティリティは、プラットフォームごとに異なるデフォルト・ローカライゼーション値を使用します。バージョン 11.1 より前のユーティリティと PC ユーティリティは、DB-Library を使用して構築されている場合があります。

すべてのユーティリティでは、ユーザが次のものに対してカスタム値を指定できます。

- 表示文字セット
- サーバ・メッセージに使用する言語
- ユーティリティが使用する文字セット

詳細については、使用してい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』を参照してください。

ヒント

この項では、国際化アプリケーションの開発と実行に関するヒントを示します。

必要なファイルがインストール済みであることの確認

いくつかの Open Client/Server ルーチンを使用する場合、特定の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必要があります。これらのファイルがインストールされていない場合、Client-Library または Server-Library はエラー・メッセージを英語で生成して、標準のエラー出力に書き込みます。

表 3-9 は、ローカライゼーション・ファイルを必要とする Open Client/Server ルーチンを示しています。

表 *3-9:* ローカライゼーション・ファイルを必要とする *Open Client/Server* ルーチン

ルーチン	必要なファイル	ファイル・ロケーション
cs_ctx_alloc	*locales.dat*	*locales/*
	objectid.dat	*ini* (Microsoft Windows の場合) *config* (UNIX の場合)
	cslib.loc	*locales/message/language_name*
	common.loc	*locales/message/language_name*
	charset.loc	*charsets/charset_name*
	binary.srt ファイル または一致するロケール・ファイル・エントリに指定されているソート・ファイル	*charsets/charset_name*
cs_locale	*charset.loc*	*charsets/charset_name*
	binary.srt ファイル または一致するロケール・ファイル・エントリに指定されているソート・ファイル	*charsets/charset_name*
ct_init	*ctlib.loc*	*locales/message/language_name*
srv_init	*srvlib.loc*	*locales/message/language_name*

Open Client/Server ルーチンでの CS_NULLTERM の使用

CS_NULLTERM がバッファ長として Client-Library ルーチン、Server-Library ルーチン、または CS-Library ルーチンに渡された場合は、そのバッファ内の値が null で終了する (1 バイトの 0 で終了する) ことを示します。

null で終了する文字列をサポートしない文字セットもあります。アプリケーションがこのようなタイプの文字セットをサポートする必要がある場合は、CS_NULLTERM を使用しないでください。

表 3-10 は、CS_NULLTERM 構造体を使用できる CS-Library ルーチン、Client-Library ルーチン、Server-Library ルーチンを示します。

表 *3-10: CS_NULLTERM* を使用する *Open Client/Server* ルーチン

ライブラリ	ルーチン	説明
CS-Library	cs_objects	オブジェクトおよびオブジェクトに関連するデータを保存、検索、クリアする。
	cs_strbuild	NULL バイトを付けないで文字セットのネイティブ言語のメッセージ文字列を構成する。
	cs_strcmp	指定されたソート順を使用して 2 つの文字列を比較する。
Client-Library	ct_connect	サーバに接続する。
	ct_cursor	カーソル・コマンドを開始する。
	ct_debug	デバッグ用のライブラリ・オペレーションを管理する。
	ct_dyndesc	動的 SQL 記述子領域でオペレーションを実行する。
	ct_labels	セキュリティ・ラベルを定義したりクリアしたりする。
	ct_options	サーバ・オプションの値の設定または検索を行う。
	ct_remote_pwd	サーバ間接続に使用するパスワードを定義またはクリアする。

ライブラリ	ルーチン	説明
Server-Library	srv_config	サーバの設定パラメータを設定する。
	srv_convert	あるデータ型から別のデータ型にデータを変換する。
	srv_createmsgq	メッセージ・キューを作成する。
	srv_createmutex	相互排他セマフォを作成する。
	srv_define_event	ユーザ・イベントを定義する。
	srv_deletemsgq	メッセージ・キューを削除する。
	srv_deletemutex	srv_createmutex を使用して作成したミューテックスを削除する。
	srv_describe	結果ロー・カラムとそのデータ・ソースを記述する。
	srv_envchange	環境の変化をクライアントに通知する。
	srv_getobjid	指定の名前のメッセージ・キューまたはミューテックスのオブジェクト ID を検索する。
	srv_getobjname	ある識別子のメッセージ・キューまたはミューテックスの名前を得る。
	srv_init	Open Server を初期化する。
	srv_log	Open Server ログ・ファイルにメッセージを書き込む。
	srv_options	オプション情報をクライアントに送信、またはクライアントから受信する。
	srv_paramnumber	現在のリモート・プロシージャ・コールのあるパラメータの位置番号を返す。
	srv_regdefine	プロシージャを登録するプロセスを開始する
	srv_regdrop	プロシージャの登録を解除する。
	srv_reginit	レジスタード・プロシージャの実行を開始する。
	srv_regnowatch	レジスタード・プロシージャの通知リストから、クライアント・スレッドを削除する。
	srv_regparam	定義されているレジスタード・プロシージャに対してパラメータを記述する、またはレジスタード・プロシージャの実行に対してデータを提供する。
	srv_regwatch	指定されたプロシージャの通知リストに、クライアント・スレッドを追加する。
	srv_returnval	非リモート・プロシージャ・コールの戻り値を定義する。
	srv_sendmsg	クライアントにメッセージを送信する。
	srv_setustate	スレッド構造体内のユーザ・ステータス・フィールドを設定する。レジスタード・プロシージャ sp_ps と sp_who を実行すると、このフィールドが表示される。
	srv_tabname	ブラウズ・モードの結果セットと対応するテーブルの名前を与える。

第 4 章

コード化文字セット変換のサポート

この章では、Open Client/Server 製品で文字セット変換がどのように行われるかについて説明します。

この章の内容は、次のとおりです。

定義

この章では次の定義を使用します。

- 「文字セット」とは、コード化を伴わない文字またはグリフの有限集合です。
- 「コード化」とは、文字セット内のそれぞれの文字を数値コードを使用してユニークに識別する処理です。
- 「コード化文字セット」とは、文字セットを表す数値コードの集合です。

  変換はコード化に依存するため、この章では「文字セット」ではなく「コード化文字セット」という用語を使用します。
- 「文字セット変換」とは、あるコード化文字セットの文字を別のコード化文字セットの文字にマップする処理です。
- 「直接変換」とは、あるコード化文字セットから別のコード化文字セットへ変換する処理です。Adaptive Server Enterprise と Open Server は、西欧言語と日本語の言語グループの文字セット間での直接変換をサポートしています。
- 「間接変換」とは、中間のコード化文字セットを経由して行われる、あるコード化文字セットから別のコード化文字セットへの変換です。

間接変換の場合、文字セットが同じ言語グループのものであるかどうかに関係なく他のどのような文字セットにも変換できるので、「汎用変換」と呼ばれることもあります。

サポートされている文字セット

注意 Open Server および SDK は、Adaptive Server Enterprise と同じ文字セットをサポートしています。

一般に Adaptive Server Enterprise と Open Client/Server 製品には、次の文字セットをサポートするファイルが付属しています。

- Apple Macintosh Roman (mac)
- IBM コード・ページ 850 (cp850)
- IBM コード・ページ 437 (cp437)
- ISO 8859-1 (iso_1)
- ISO 8859_15 (iso_15:Latin9 – 西欧)
- Hewlett-Packard Roman 8 と Roman9 (roman8 と roman9)
- UTF-8 形式でコード化された Unicode (utf8)
- GB18030-2000 標準に準拠する中国語
- 韓国語コード・ページ 949 (cp949)
- カザフ語 (kz1048)

日本語の言語モジュール製品には、次の文字セットをサポートするためのファイルが含まれています。

- DEC Kanji (deckanji)
- EUC JIS (eucjis)
- シフト JIS (sjis)

サポートされている言語と文字セットの完全なリストについては、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。

コード化文字セット変換の概要

文字セット変換を行うことによって、さまざまなコード化文字セットを使用するクライアントとサーバが通信できます。

現在の Sybase システムでは、サーバでのみ自動文字セット変換が実行されます。Adaptive Server Enterprise と Open Server は、西欧言語と日本語の言語グループの文字セット間での直接コード化文字セット変換をサポートしています。これらの文字変換が、Adaptive Server Enterprise と Open Server がサポートする唯一の直接文字セット変換です。ただし Open Server は、UTF-8 形式の Unicode の文字セットと Sybase がサポートする文字セットとの間の双方向の変換をサポートしています。これによって、Open Server は任意の 2 つの Sybase 文字セット間の間接変換 (charset_1 → Unicode → charset_2) を実行できます。

Unicode 標準は、ISO 10646 標準と同じ国際文字セットです。Unicode は、世界の主要な書き言葉で使用されるすべての文字を仮想的にコード化します。

UTF-8 は、ストリームベースのアプリケーションと互換性がある Unicode のマルチバイトの可変長コード化です。X/Open 標準、POSIX 標準、X11 標準でのデータ交換とデータ記憶に対する使用をおすすめします。

接続に対する言語と文字セットの確立

クライアント・アプリケーションは、サーバに接続しようとするときに次の内容を指定する接続要求を送信します。

- その接続の文字セット変換を無効にするかどうか (Client-Library の CS_NOCHARSETCNV プロパティまたは DB-Library の DBSETLCHARSET ルーチンを使用します)
- その接続に対して使用する文字セット
- その接続に対して使用する言語

サーバは要求された言語と文字セットをサポートできるかどうかを調べてから、その接続を受け入れます。

表 4-1 は、接続時の Adaptive Server Enterprise と Open Server の動作の概要を示します。

表 *4-1:* クライアントとサーバの変換動作

サーバがクライアントの文字セットをサポートするか	サーバがクライアントの言語をサポートするか	サーバの動作	ct_connect	dbopen
はい	はい	クライアントの言語と文字セットでその接続を受け入れる。	CS_SUCCEED を返す。	SUCCEED を返す。
いいえ	はい	文字セット変換が無効になっている場合はその接続を受け入れるが、強制的にその接続がサーバ自体の文字セットを使用するよう設定する。	CS_SUCCEED を返す。	SUCCEED を返す。
		文字セット変換が無効になっていない場合は、その接続を拒否する。	CS_FAIL を返す。	FAIL を返す。
はい	いいえ	その接続が次の言語と文字セットを使用することをクライアントに通知する。 • us_english 言語 • クライアントの文字セット	CS_SUCCEED を返す。	SUCCEED を返す。
いいえ	いいえ	文字セット変換が無効になっている場合はその接続を受け入れるが、強制的にその接続が次の言語と文字セットを使用するよう設定する。 • us_english 言語 • サーバ自体の文字セット	CS_SUCCEED を返す。	SUCCEED を返す。
		文字セット変換が無効になっていない場合は、その接続を拒否する。	CS_FAIL を返す。	FAIL を返す。

接続が確立されると、サーバは次のタスクを行います。

- その接続でネゴシエートした言語と文字セットですべてのメッセージを生成する
- (その接続に対して文字セット変換が無効になっていない場合は) 受信データと送信データの両方に対して必要なすべての文字セット変換を実行する

文字セット変換の無効化

クライアント・アプリケーションは、一般に次のどちらかの理由があると、文字セット変換を無効にします。

- クライアント・アプリケーションはサーバがサポートしていない文字セットでデータを保管、検索する必要があるため。
- 必要な文字セット変換はすべてクライアント・アプリケーションが実行するため。

文字セット変換が無効になっている場合、Adaptive Server Enterprise は Transact-SQL® 文、プロシージャ、テーブル、ビューとその他の名前、またはデータでは文字セット変換を実行しません。サーバは次のように動作します。

- Transact-SQL 文と名前が標準の Transact-SQL であると仮定します。
- データ値を送信されたとおりに正確に保管します。
- デフォルトの文字セットでメッセージを生成します。

Client-Library アプリケーションは CS_NOCHARSETCNV 接続プロパティを CS_TRUE に設定して接続の文字セット変換を無効にしてから、ct_connect を呼び出してその接続をオープンできます。

DB-Library アプリケーションは *char_set* パラメータに NULL を指定して DBSETLCHARSET を呼び出して接続の文字セット変換を無効にしてから、dbopen を呼び出してその接続をオープンできます。

変換ゲートウェイとしての Open Server の使用

Open Server は Sybase がサポートするすべての文字セットと Unicode (ISO 10646 標準と同等)、UTF-8 との間で双方向に変換できるので、Open Server アプリケーションは Sybase がサポートする任意の 2 つの文字セット間での間接変換を実行できます。このため、Open Server アプリケーションを使用した場合に、異なる言語グループの文字セットを使用するアプリケーションとサーバの間の通信が可能になります (データを失う可能性もあることに注意してください)。

変換ゲートウェイとしての Open Server アプリケーションの設定方法の詳細については、「Open Server ゲートウェイに対してローカライズされた接続の作成」(27 ページ) を参照してください。

文字セット変換中に使用されるファイル

この項では、文字セット変換中に使用されるファイルについて説明します。

Unilib ライブラリ

Unilib® ライブラリ、*libsybunic* には、UTF-8 形式の Unicode (ISO 10646 標準と同等) 文字セットと Sybase がサポートする文字セットとの間の双方向の変換をサポートする Unicode ベースのルーチンが含まれています。

カスタムのコード化文字セット変換の使用

Open Server を使用した場合、アプリケーションはカスタム変換ルーチンをインストールできます。カスタム変換ルーチンがインストールされると、指定されたタイプの変換が要求された場合、Open Server は自動的にそれらのカスタム変換ルーチンを使用します。

カスタム変換ルーチンをインストールする目的

Open Server に付属している変換機能では十分でない場合は、カスタム文字セット変換ルーチンをインストールしてください。カスタム変換ルーチンをインストールする主な理由は、間接的な変換を直接的な変換に置き換えることでパフォーマンスを改善させることです。

たとえば、Open Server アプリケーションで ISO 8859-1 と EUC JIS の間の変換を実行するためのカスタム・ルーチンをインストールするとします。この直接変換は、Open Server で提供される間接変換 (ISO 8859-1 → Unicode UTF-8 → EUC JIS または ISO 8859-1 ← Unicode UTF-8 ← EUC JIS) よりも高速である場合があります。

カスタム変換ルーチンの作成

カスタムの文字セット変換のカスタム・ルーチンは次のように定義します。

```
CS_RETCODE convfunc(context, srcfmt, srcdata,
        destfmt, destdata, destlen)
CS_CONTEXT      *context;
CS_DATAFMT      *srcfmt;
CS_VOID         *srcdata;
CS_DATAFMT      *destfmt;
CS_VOID         *destdata;
CS_INT          *destlen;
```

各パラメータの意味は、次のとおりです。

- *context* は CS_CONTEXT 構造体へのポインタです。
- *srcfmt* は、変換元データを記述する CS_DATAFMT 構造体へのポインタです。*srcfmt→maxlength* は、変換元データの実際の長さをバイト単位で記述します。
- *srcdata* は変換元データへのポインタです。

- *destfmt* は、変換先データを記述する CS_DATAFMT 構造体へのポインタです。*destfmt→maxlength* は、変換先データ・スペースの実際の長さをバイト単位で記述します。
- *destdata* は変換先データ領域へのポインタです。
- *destlen* は整数へのポインタです。変換が成功した場合、カスタム・ルーチンは **destlen* を **destdata* に指定されるバイト数に設定します。

cs_config は、カスタム変換ルーチンから呼び出すことができる唯一の CS-Library ルーチン、Client-Library ルーチン、または Server-Library ルーチンです。

カスタム・ルーチンが CS_SUCCEED 以外の値を返すと、CS-Library は CS-Library エラーを表示します。CS-Library がどのようなタイプのエラーを表示するかは、カスタム・ルーチンがどのような値を返すかによって決まります。

表 4-2 は、カスタム変換ルーチンに対して有効な戻り値を示します。

表 *4-2:* カスタム変換ルーチンの戻り値

戻り値	意味
CS_SUCCEED	変換が成功した。
CS_TRUNCATED	変換の結果がトランケートされた。
CS_MEM_ERROR	メモリの割り付け障害が発生した。
CS_EBADXLT	変換できない文字があった。
CS_ENOXLT	要求された変換がサポートされていない。
CS_EDOMAIN	変換元の値がデータ型の正しい値のドメインの範囲外である。
CS_EDIVZERO	0 による除算が許可されていない。
CS_EOVERFLOW	変換の結果がオーバフローした。
CS_EUNDERFLOW	変換の結果がアンダフローした。
CS_EPRECISION	変換の結果、精度が損なわれた。
CS_ESCALE	無効な位取り値が検出された。
CS_ESYNTAX	変換結果が、変換先のタイプに対して構文的に正しくない値になった。
CS_ESTYLE	スタイル・エラーのために変換オペレーションが停止した。

カスタム変換ルーチンのインストール

アプリケーションは cs_manage_convert を呼び出して、カスタム変換ルーチンをインストールします。cs_manage_convert の詳細については、『Open Client/Server Common Libraries リファレンス・マニュアル』を参照してください。

リリース 4.9 より前の Adaptive Server Enterprise での文字セット変換

4.9 より前のリリースでは、Adaptive Servers Enterprise データ・サーバは、文字セットの変換を行いません。クライアント・アプリケーションがリリース 4.9 より前の Adaptive Server Enterprise と通信して、サーバとは異なる文字セットを使用する場合には、国際文字が正しく表現されない可能性があります。

この問題を解決するには次のような方法があります。

- クライアント・アプリケーションの文字セットを変更して、Adaptive Server Enterprise の文字セットと一致させます。
- cs_manage_convert を使用してカスタムの文字セット変換ルーチンをインストールし、cs_convert を呼び出してデータを変換してから、サーバにそのデータを送信します。

メインフレームのサポート

メインフレーム・システムは一般に状態を持つ文字コードを使用します。このメカニズムでは、エスケープ文字があると文字列の途中で別の文字コードに変わります。

Open Client/Server 製品は、このメカニズムをサポートしていません。

第 5 章

ロケール・ファイルの編集

この章では、ロケール・ファイルの概要とその変更方法について説明します。

この章の内容は、次のとおりです。

ロケール・ファイルの名前は *locales.dat* です。このファイルは Sybase ディレクトリ・ツリーの *locales* サブディレクトリに格納されています。「付録 A 国際化に関連するディレクトリとファイル」を参照してください。

処理の概要

この項では、ロケール定義の追加または変更処理の概要について説明します。ロケール・ファイルとそのファイルの編集方法の詳細については、このあとの項を参照してください。

❖ **ロケール定義の追加または変更を行うには、次の手順に従います。**

1. 編集後のバージョンに問題が発生した場合のことを考慮して、*locales* ディレクトリにあるロケール・ファイル (*locales.dat*) のコピーを作成します。
2. ロケール・ファイルを編集します。つまり、プラットフォームごとの適切なセクションで新しくエントリを追加するか既存のエントリを変更します。
3. ローカライゼーション環境変数 (LC_ALL、LC_CTYPE、LC_MESSAGE、LC_TIME、LANG) を適切に更新します。
4. 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。

ロケール・ファイルの編集

あらかじめ定義されているロケール・ファイル・エントリでは十分でない場合は、それらのエントリを変更したり、新しいロケール名を定義するエントリを追加したりすることもできます。たとえばロケール・ファイルを編集して、次のタスクを行う必要がある場合もあります。

- ロケール・エントリで指定されている言語、文字セット、または照合順を変更すること。
- 新しい言語モジュールに必要なロケール定義などを追加すること。
- Sybase 以外のソフトウェアで使用されるロケール名と一致させること。たとえば、次のように Sybase であらかじめ定義されている 1 つのロケール名が “fr” であるとします。

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

  Sybase 以外のアプリケーションで LC_ALL 環境変数値が “french” のときに、Open Client/Server アプリケーションで LC_ALL 環境変数を使用して、このロケール・ファイル・エントリでローカライズするように設定する場合は、新しいエントリを追加するか、既存のエントリで指定されているロケール名を次のように変更する必要があります。

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

ロケール・ファイルのセクションとエントリ

ロケール・ファイルは Sybase リリース・ディレクトリの *locales* サブディレクトリの下にあります。

ロケール・ファイルは次のセクションから構成されています。

- 標準のセクション (表 B-2 (77 ページ) を参照)
- ロケール定義エントリが指定されている、プラットフォームに依存するセクション

ロケール定義エントリ

ロケール・ファイルにはプラットフォームに依存するセクションがあり、これらの各セクションは、あらかじめ定義されているロケール定義エントリから構成されます。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]
```

各パラメータの意味は、次のとおりです。

- *locale_name* には、ロケール定義の名前が入ります。*locale_name* は通常ベンダが指定するものであり、POSIX 用語に準拠しています。ロケール・ファイルの最後にあるコメントには、ロケール名の POSIX 値がリストされます。
- , (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* には、Sybase 製品が照合順を認識するのに使用するファイル名が入ります。*sortorder_name* はオプションです。sortorder_name を指定しない場合、Open Client/Server 製品はバイナリの照合順を使用します。

次のロケール・ファイル・エントリはフランス語のロケールを指定します。このロケールではソート順が指定されていないので、デフォルトのソート順である “binary” が使用されます。

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

ロケール・ファイルの例

次の例は、ロケール・ファイル内でプラットフォームに依存するセクションの一部を示したものです。

```
[aix]
      locale = en_US, us_english, iso_1
      locale = en_US.ISO8859-1, us_english, iso_1
      locale = en_JP, us_english, eucjis
      locale = FR_FR.IBM-850, french, cp850
      locale = fr_FR.ISO8859-1, french, iso_1
      locale = fr_CA, french, iso_1
      locale = Fr_CA.IBM-850, french, cp850
      locale = fr_CA.ISO8859-1, french, iso_1
```

```
[linux]
      locale = GERMAN, german, iso_1
      locale = de, german, iso_1
      locale = de_AT, german, iso_1
      locale = de_AT.437, german, cp437
      locale = de_AT.850, german, cp850
      locale = CHINESE, chinese, eucgb
      locale = zh_CN, chinese, eucgb
      locale = zh_CN.GB18030, chinese, gb18030
      locale = zh_CN.gbk, chinese, eucgb
      locale = zh_TW, tchinese, big5
```

ロケール・ファイルの編集

次の作業を行ってから、ロケール・ファイルを編集してください。

- 使用しているプラットフォーム用のセクションにリストされているエントリを探して、適切なエントリがあるかどうかを調べてください。適切なエントリが存在していれば、ロケール・ファイルを編集する必要はありません。
- 編集後のバージョンで問題が発生した場合のことを考慮して、オリジナルのロケール・ファイルのバックアップ・コピーを作成しておいてください。

エントリの追加または変更

ロケール・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したりするには、次の手順に従ってください。

1 *locale_name* に使用する値を選択します。

locale_name にはどのような値でも指定できます。*language.territory* という形式の名前を選択することをおすすめします。

2 *language_name* に使用する値を決定します。

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* に使用する値を決定します。

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

4　(バイナリ以外のソート順を選択する場合は) *sortorder_name* に使用する値を決定します。

charsets/*charset_name* サブディレクトリには、文字セットのソート順ファイル (*.srt*) があります。*sortorder_name* は、*.srt* 部分を除いて、これらのファイル名のうちの 1 つと一致している必要があります。

5　ロケール・ファイル内のプラットフォームに依存する適切なセクションで、適切なエントリを新しく入力するか既存のエントリを変更します。

変更後は、次の作業を行ってください。

- ローカライゼーション環境変数 (LC_ALL、LC_CTYPE、LC_MESSAGE、LC_TIME、LANG) を適切に更新します。
- 新しいロケール名をすでに追加しており、既存のアプリケーションの 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。

エントリの削除

アプリケーションが特定のエントリを使用しなくなった場合でも、ロケール・ファイルからそのエントリを削除する必要はありません。ただし、エントリの削除を決定した場合には、そのエントリを使用するアプリケーションがないことを確認してください。

第 6 章

照合順の作成または変更

この章では、照合順 (ソート順) ファイルの作成および変更方法について説明します。

この章の内容は、次のとおりです。

処理の概要

この項では、ソート順ファイルの作成および変更処理の概要について説明します。詳細については、このあとの項を参照してください。

❖ **ソート順ファイルの作成または変更を行うには、次の手順に従います。**

1 出荷時に提供されている **.srt* ファイルの 1 つをコピーし、*.srt* サフィックスの部分はそのままにしてファイル名を変更します。

注意 製品に付属している **.srt* ファイルは変更しないでください。代わりに、オリジナルの **.srt* ファイルのコピーを作成して、そのコピーを変更してください。

---

2 新しく作成されたファイルを編集して、次のようにして既存のエントリを変更するかまたは新しいエントリを追加します。

- “class”、“id”、“menuname”、“charset”、“preference”、“description” など、[sortorder] セクションの一般的なエントリを指定します。
- “lig = *value*” というエントリ形式を使用して、合字をリストします。文字エントリよりも前に合字エントリをグループ分けします。
- “char = *value*” というエントリ形式を使用して、すべての文字セットの文字とグリフを適切なプライマリ・ソート順でリストします。

- セカンダリ・ソート順については、“char = *value1, value2, ...*” というエントリ形式を使用して、プライマリ・ソート順のエントリの横に値を追加していきます。
- 大文字と小文字を区別しないソートについては、対応する大文字と小文字を等号で結合します。

3 新しい *.srt* ファイルを *charset_name* サブディレクトリの下の *charsets* ディレクトリに保存します。

4 適切にロケール・ファイル・エントリを編集して、新しい照合順ファイルを参照するようにします。

照合順の概要

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。

照合順の定義は文字セット定義に基づいて構築されますが、同じ文字セットを使用する言語でも、文字は異なる順序でソートされる場合があります。たとえば、スペイン語では “Ch” は単一の文字とみなされるために “Co” が “Cho” よりも前になりますが、英語のアルファベット順では “Cho” は “Co” よりも前になります。

順序規則は、文字とアクセント記号の組み合わせによって言語間で異なる場合もあります。たとえば、“a” (アクセント記号なし) は “b” よりも前ですが、“Å” は “z” の後になる場合があります。

この項では、照合順を定義する場合に一般に考慮する点について説明しますが、必要な情報がすべて示されているわけではありません。照合順についての一般的な資料を参照してください。

定義

Sybase の照合順についてまだ理解していない場合は、次の定義を参考にしてください。

- 照合順の「プライマリ・ソート順」は、“char=” で始まる行の縦方向の順序です。
- プライマリ・エントリの「セカンダリ・ソート順」は、1 つの “char =” 行にある文字の横方向の順序です。

ソートのタイプ

文字をソートするには多くの方法があります。Open Client/Server の照合順ファイルでは、表 6-1 にリストされているソート・タイプのうちの 1 つが使用される場合もあれば、複数のソート・タイプが使用される場合もあります。

表 *6-1:* ソート順序のタイプ

ソート・タイプ	説明
1 段階ソート	文字はプライマリ・ソート順の値に従ってソートされる。 “char=” エントリの縦方向のリストで、上の行にある文字は下の行にある文字よりも前にソートされる。
2 段階ソート	文字はプライマリ・ソート順とセカンダリ・ソート順の値に従ってソートされる。2 つの文字列内のすべての文字がプライマリ・ソート値と同じ場合、文字のセカンダリ・ソート値は、1 段階ソートでは区別がつかない部分のソートを行うために使用される。 2 つの文字が同じ “char =” 行に表示されている場合は、一番左側にある文字が最初にソートされる。 たとえば次のような内容のソート順ファイルがあると仮定する。 char = A,a,Ä,ä char = B,b char = C,c,Ç,ç 上記の文字を使用するいくつかの文字列は、次のようにソートされる。 ABC ÄBC äbc acb ächb 文字列 ABC、ÄBC、鉤 c のプライマリ値は同じなので、セカンダリ・ソート値に従って順序付けられる。acb と 劃 b はセカンダリ値に従って同様にソートされる。b は c よりもプライマリ値が前に位置付けられているので、鉤 c は acb よりも前にくる。
1 文字を 2 文字とみなすソート	複数の文字としてソートされる単一の文字は「合字」と呼ばれる。たとえばドイツ語の文字 “ß” は “ss” としてソートされる。
2 文字を 1 文字とみなすソート	1 つの文字としてソートされる 2 つの文字列は「ソート・ダブル」と呼ばれる。たとえばスペイン語の文字列 “ch” は、“c” と “d” の間にくる 1 つの文字としてソートされる。

大文字と小文字の区別の決定

大部分の照合順ファイルでは、単一の文字のすべての変形が 1 つの char = 行にリストされます。

大文字と小文字を区別する照合順では、ある文字の大文字と小文字の変形がソートされる順序でリストされ、それぞれの文字はカンマで区切られます。次に例を示します。

```
char = 0x41,0x61,0xC0,0xE0,0xC1,0xE1,0xC2,0xE2,0xC3,0xE3
;A, a, A-grave, a-grave, A-acute, a-acute, A-tilde, a-tilde,
;A-diaeresis, a-diaeresis
;
char = 0x42,0x62
;letter B, b
```

大文字と小文字を区別しない照合順では、ある文字の大文字と小文字の変形が任意の順序でリストされて、それぞれの文字が等号で結合されます。次に例を示します。

```
char = 0x41=0x61,0xC0=0xE0,0xC1=0xE1,0xC2=0xE2,0xC3=0xE3
;A, a, A-grave, a-grave, A-acute, a-acute, A-tilde, a-tilde,
;A-diaeresis, a-diaeresis
;
char = 0x42=0x62
;letter B, b
```

カスタム照合順ファイルの作成

大部分のプラットフォームでは、Open Client/Server 製品には表 6-2 のような標準の照合順ファイルが含まれています。

表 *6-2:* 一般に提供されている照合順ファイル

ファイル名	説明
binary.srt	順序はそれぞれの文字の内部バイナリ値と対応している。*binary.srt* にはエントリ “binary = true” が指定されている。 このソート順ではローカライゼーション・ファイルは必要ない。
dictionary.srt	辞書順であり、大文字と小文字は区別される。対応する小文字よりも前にくる大文字に対してはプライマリ辞書式順が適用される。アクセント記号付きの文字に対してはセカンダリ順が適用される。 ファイル名は言語によって異なる。たとえばスペイン語バージョンの場合は *espdict.srt* というファイル名になる。
noaccents.srt	辞書順 (アクセントを区別しない、大文字と小文字を区別しない)。アクセント記号なしの文字で始まるワードと、対応するアクセント記号付きの文字で始まるワードが混在している。 ファイル名は言語によって異なる。たとえばスペイン語バージョンの場合は *espnoac.srt* というファイル名になる。
nocase.srt	辞書順であり、大文字と小文字は区別されない。大文字で始まるワードと、対応する小文字で始まるワードが混在している。 ファイル名は言語によって異なる。たとえばスペイン語バージョンの場合は *espnocs.srt* というファイル名になる。
nocasepref.srt	辞書順であり、大文字と小文字は区別されない。同じ文字の小文字がある場合にかぎり大文字が優先される。

使用している言語にさらに照合順要件がある場合は、「照合順ファイルの概要」(55 ページ) のガイドラインに従って、カスタム照合順ファイルを作成できます。

照合順ファイルの概要

照合順ファイルは **.srt* という名前であり、*charsets/charset_name/* ディレクトリにあります。すべての照合順ファイルは、標準の Sybase 外部ローカライゼーション・ファイルの構文を使用します。

「付録 B 外部ローカライゼーション・ファイルの構文」を参照してください。

照合順ファイルのセクションとエントリ

照合順ファイルには次のような要素があります。

- コメント行、版権セクション、ファイル・フォーマット・セクションの詳細については、表 B-2 (77 ページ) を参照してください。
- 一般エントリの詳細については、表 6-4 (59 ページ) を参照してください。
- 合字エントリの詳細については、「カスタム照合順ファイルの作成」(58 ページ) の手順 3 を参照してください。
- 文字エントリの詳細については、「カスタム照合順ファイルの作成」(58 ページ) の手順 4、5、6、7 を参照してください。

照合順ファイルへの文字の書き込み

照合順ファイル・エントリに文字を書き込むには、次の 3 とおりの方法があります。

- 文字の 16 進数文字コード化を入力する方法。次に例を示します。

```
char = 0x20 ;( ) space
char = 0x3D ;(=) equals sign
```

- 文字を引用符で囲んで入力する方法。次に例を示します。

```
char = " " ;( ) space
char = "=" ;(=) equals sign
```

- 文字を引用符で囲まないで入力する方法。次に例を示します。

```
char = A, a
char = B, b
```

表 6-3 は、照合順ファイル・エントリへの書き込み方法に応じて文字を分類して示します。

表 *6-3:* 照合順ファイル・エントリへの文字の書き込み

文字のタイプ	16 進数で書き込み可能か	引用符を使用して入力可能か	引用符を使用せずに入力可能か
印刷不可能な文字とキーボード上にない文字	はい	いいえ	いいえ
スペース (“ ”) 等号 (“=”) コメント文字 エスケープ文字 リスト・セパレータ文字	はい	はい	いいえ
円記号 (“¥”)	はい	はい。ただし引用符の内部には円記号を 2 つ入力する必要がある (“¥¥”)。	いいえ
その他のすべての文字	はい	はい	はい

preference キーワードと *order by* 句

大文字と小文字を区別しない照合順ファイルでは、preference エントリを使用して、order by 句を指定した select 文の結果としてソート出力が生成されたときに、等号の左側にある文字が右側にある文字よりも前にソートされるかどうかを示すことができます。

たとえば照合順ファイルに次のようなエントリがあると仮定します。

```
char = A=a, Á=á
char = B=b
```

preference=true となっている場合は、order by 出力では次のようにソートされます。

```
Aab
aAb
Aáb
```

preference=false となっている場合は、order by 出力では次のどちらかの順序でソートされる可能性があります。

```
aAb
Aab
Aáb
```

または

```
Aab
aAb
Aáb
```

preference キーワードの特徴は次のとおりです。

- 大文字と小文字を区別しないソート順だけに適用されます。
- order by 句を指定した結果として行われるソートだけに影響します。

preference=true となっている場合は、等号の左側にある文字が最初にソートされます。preference=false となっている場合は、等号の左側にある文字は最初にソートされないことがあります。

preference キーワードのデフォルト値は “true” です。つまり照合順ファイルに preference エントリが記述されていない場合には、order by 句を指定してソートを実行すると、等号の左側にある文字に優先度が設定されます。

一般に preference=true となっている場合は、大文字が小文字よりも前にソートされることを意味します。

カスタム照合順ファイルの作成

この項では、カスタム照合順ファイルの作成方法について説明します。この項の説明をすべて読んで Open Client/Server 製品に含まれている照合順ファイルについてよく理解してから、カスタム照合順ファイルの作成を開始してください。

照合順ファイルの例については、「照合順ファイルの例」(62 ページ) を参照してください。

ローカライゼーション・ファイルの構文の詳細については、「付録 B 外部ローカライゼーション・ファイルの構文」を参照してください。

照合順ファイルの作成または変更を行うには、次の手順に従ってください。

1 出荷時に提供されている *.srt* ファイルをモデルとして使用する場合は、オリジナルのファイルを上書きしないように必ずこのファイルをコピーして名前を変更します。新しいファイルの名前には *.srt* サフィックスを指定する必要があります。また、ファイル名のうちサフィックスを除いた部分に、サポートする言語とそのファイルに関連した説明的な名前を付けると便利です。

2　一般的なエントリの値を決定します。表 6-4 では、これらの一般的なエントリについて説明します。

表 *6-4: .srt* ファイルの一般的なエントリ

エントリ・キーワード	説明	必須	エントリ値
class	ソート順クラス 現時点でサポートされているクラスは、8 ビット文字セットのクラス 1 だけである。	はい	0x01d
id	照合順を識別するユニークな 16 進数。	はい	ユーザ定義の照合順の場合、ID は 0xC9 から 0xFF までの間の値である必要がある。 0x00 から 0xC8 までの間の 16 進数は Sybase によって予約されている。
menuname	sybinit プログラムに表示されるような照合順の名前。	はい	推奨値は 64 文字以内の文字列。文字列は、sybinit プログラムによって 64 文字にトランケートされる。 これはユーザ定義値である。
name	照合順の名前。	いいえ	30 文字以内の文字列。 これはユーザ定義値である。
charset	この照合順ファイルが適用される文字セット。 これは、この照合順ファイルが置かれているディレクトリの名前でもある。	はい	この値は、Sybase ディレクトリ・ツリー内の文字セットのサブディレクトリ名と一致する必要がある。
preference	大文字と小文字を区別しないソート順の場合、order by 句を指定した select 文の結果としてソート出力が生成されたときに、等号の左側にある文字に優先度を付けるかどうかを指定。	いいえ	false − 優先度を付けない。 true − 等号の左側にある文字に優先度を付ける。値 “true” は “false” よりもパフォーマンスに与える影響が大きい。 デフォルトは “true”。
description	照合順について記述する説明文。照合順とともに保管される。	いいえ	255 文字以内の文字列。 これはユーザ定義値である。

3 合字があるかどうかを調べます。「合字」とは、複数の文字として格納される単一の文字です。合字がある場合は、次のようにします。

- 合字 (“lig”) エントリをまとめて “char” エントリの前に置きます。
- 適用できる場合は、合字の大文字と小文字の形を両方とも指定します。

大文字と小文字を区別する合字の構文は次のようになります。

```
lig = value, after characters ;case-sensitive sort
```

ここで、

- *characters* は、ソートされる合字の前の文字を表す文字列です。
- *value* は、合字の 16 進数コード化、引用符なしの合字、または引用符付きの合字です。

大文字と小文字を区別しない合字の構文は次のようになります。

```
lig = value1=value2, after characters ;case-insensitive
sort
```

ここで、

- *value1* と *value2* は、大文字と小文字の合字の 16 進数コード化、引用符なしの合字、または引用符付きの合字です。
- *characters* は、ソートされる合字の前の文字を表す文字列です。

次の例は、大文字と小文字を区別しない ISO 8859-1 用の照合順ファイル内の合字エントリを示します。

```
lig = 0xC6, after AE ;diphthong AE, A with E
lig = 0xE6, after ae ;diphthong ae, a with e
char = 0x41,0x61,0xC0,0xE0,0xC1,0xE1,0xC2,0xE2x
;varieties of letter A
char = 0x42,0x62 ;B, b
. . .
```

4 ソート順のすべての文字エントリを縦方向にリストします。この縦方向のリストがプライマリ・ソート順です。

文字エントリの構文は次のようになります。

```
char = value
```

ここで、*value* はその文字の 16 進数のコード化セット・コード、引用符なしの文字、または引用符付きの文字です。

次に例を示します。

```
char = 0x41    ;ISO 8859-1 code set.
```

5　適用できる場合は、次のようにセカンダリ・ソート順の情報をファイルに追加します。

- 大文字と小文字を区別するソート順の場合 (大文字に優先度を付ける場合)、大文字の右側に小文字の変形を指定します。リスト・セパレータ文字を使用して文字を区切ります。
- 大文字と小文字を区別しないソート順では、(アクセント記号付きの文字を含む) それぞれの大文字と小文字の対の間を等号で結合します。
- ある文字とその変形を相対的な順序関係で指定します。たとえば、フランス語の “é” を “e” の右側に指定します。これらの文字が合字でないことを確認します。つまり、プライマリ・ソート順のエントリを区切ります。その場合は、リスト・セパレータ文字を使用して文字の変形を区切ります。

次の例は、ラテン・アルファベットで大文字と小文字を区別するソート順の場合のセカンダリ・ソート順の情報を示します。

```
char = 0x41,0x61,0xC0,0xE0,0xC1,0xE1,0xC2,0xE2,
0xC3,0xE3,0xC4,0xE4,0xC5,0xE5
;A, a, A-grave, a-grave, A-acute, a-acute,
;A-circumflex, a-circumflex, A-tilde, a-tilde,
;A-diaeresis, a-diaeresis, A-ring, a-ring
. . .
char = 0x4E,0x6E,0xD1,0xF1 ;N, n, N-tilde, n-tilde
. . .
```

6　ソート・ダブルがあるかどうかを調べます。「ソート・ダブル」または「二重音字」とは、単一の文字としてソートされる一対の文字です。ソート・ダブルがある場合は、次のようにします。

- それぞれのソート・ダブルを別々の “char” エントリとしてリストします。
- 大文字と小文字を区別するソートの場合は、ソート・ダブルのすべての組み合わせを適切なソート順に設定します。

ソート・ダブルの構文は次のようになります。

```
char = value1value2
```

ここで、

- *value1* はソート・ダブルの対の最初の文字です。
- *value2* はその対となる 2 番目の文字です。

value1 と *value2* が 16 進数で書き込まれている場合は、*value1* には先行の ‘0x’ を使用しますが、*value2* には先行の ‘0x’ を使用しません。次に例を示します。

```
char = 0x4348,0x4368,0x6348,0x6368 ;CH,Ch,cH,ch
```

value1 と *value2* には引用符なしの文字を指定することも、引用符付きの文字を指定することもできます。次に例を示します。

```
char = CH, Ch, cH, ch
```

または

```
char = "CH", "Ch", "cH", "ch"
```

次の例は、iso_1 (ISO 8859-1) 文字セット用の大文字と小文字を区別する *.srt* ファイルでのスペイン語のソート・ダブル “ch” の配置を示します。

```
char = 0x41,0x61,0xC0,0xE0,0xC1,0xE1,0xC2,0xE2
;varieties of letter A
char = 0x42,0x62 ;B, b
char = 0x44,0x64,0xC7,0xE7 ;C, c, C-cedilla, c-cedilla
char = 0x4348,0x4368,0x6348,0x6368 ;CH,Ch,cH,ch
. . .
```

7 印刷不可能な文字、キーボード上にない文字、記号、言語様式に関係のある文字などその他のすべての文字を縦方向のリストに入れます。“char” エントリまたは “lig” エントリを適切に使用します。必ず “char” エントリの前にすべての “lig” エントリをまとめてグループ化します。

照合順ファイル内に非アルファベット文字を書き込む方法の詳細については、表 6-3 (57 ページ) を参照してください。

8 新しい *.srt* ファイルを *charset_name* サブディレクトリの下の *charsets* ディレクトリに保存します。

9 適切にロケール・ファイル・エントリを編集して、新しい照合順ファイルを参照するようにします。「第 5 章 ロケール・ファイルの編集」を参照してください。

照合順ファイルの例

この項では、大文字と小文字を区別する照合順ファイルの例を示します。

実際の照合順ファイルは、*charsets/charset_name/*.srt* という名前で Sybase ディレクトリ・ツリーに存在します。

```
; semi-colon is the comment character
 [sortorder]
 ;=================================================================
 ;
 ; @(#)dictionary.srt
 ;
 ; Sort Order Overview:
 ; --------------------
 ; Based on the ISO 8859-1 ("Latin 1") character set, this sort order is
 ; a case-sensitive ordering.Upper case letters always sort before their
```

```
 ; lower case counterparts.
 ;
 ; It is useful for at least the English, French and German languages,
 ; and may work for many others.
 ;
; Ligatures, Sort-Doubles, etc.:
 ; ------------------------------
 ; AE, ae ligatures
 ; German sharp-s ligature with "ss"
 ;
 ; The ordering:
 ; --------------
 ; first all non-alphanumeric characters in binary order
 ; followed by all numeric digits
 ; then all alphabetic characters used in English, French and German
 ; and ended by all alphabetic characters not used in English, French
 ; or German
 ;===========================================================================
 class = 0x01 ; Class `1' sort order
 id = 0x33 ; Unique ID # (51) for the sort order
 name = dictionary_iso_1
 menuname = "General purpose dictionary ordering."
 charset = iso_1
 description = "General purpose dictionary sort order for use with several
 Western-European languages including English, French, and German.Uses the
 ISO 8859-1 character set and is case-sensitive."
 ;
 ; ligatures for English, French, and German
 lig = 0xC6, after AE  ;AE ligature
 lig = 0xE6, after ae ;ae ligature
 lig = 0xDF, after ss ;small german letter sharp s
 ;
 ; Control characters
 char = 0x01 ;(SOH) start of heading
 char = 0x02 ;(STX) start of text
 char = 0x03 ;(ETX) end of text
 char = 0x04 ;(EOT) end of transmission
 char = 0x05 ;(ENQ) enquiry
 char = 0x06 ;(ACK) acknowledge
 char = 0x07 ;(BEL) bell
 char = 0x08 ;(BS) backspace
 char = 0x09 ;(HT) horizontal tab
 char = 0x0A ;(LF) newline, or line feed
 char = 0x0B ;(VT) vertical tab
 char = 0x0C ;(FF) form feed
 char = 0x0D ;(CR) carriage return
 char = 0x0E ;(SO) shift out
 char = 0x0F ;(SI) shift in
 char = 0x10 ;(DLE) data link escape
 char = 0x11 ;(DC1) device control 1
 char = 0x12 ;(DC2) device control 2
```

```
char = 0x13 ;(DC3) device control 3
char = 0x14 ;(DC4) device control 4
char = 0x15 ;(NAK) negative acknowledge
char = 0x16 ;(SYN) synchronous idle
char = 0x17 ;(ETB) end transmission blk
char = 0x18 ;(CAN) cancel
char = 0x19 ;(EM) end of medium
char = 0x1A ;(SUB) substitute
char = 0x1B ;(ESC) escape
char = 0x1C ;(FS) file separator
char = 0x1D ;(GS) group separator
char = 0x1E ;(RS) record separator
char = 0x1F ;(US) unit separator
;
; All non-alphanumeric characters, including puntuation.
; These are sorted by their numerical ordering, based on the
; ISO 8859-1 standard, for clarity and consistency.
;
char = 0x20 ;( ) space
char = 0x21 ;(!) exclamation mark
char = 0x22 ;(") quotation mark
char = 0x23 ;(#) number sign
char = 0x24 ;($) dollar sign
char = 0x25 ;(%) percent sign
char = 0x26 ;(&) ampersand
char = 0x27 ;(') apostrophe
char = 0x28 ;(() left parenthesis
char = 0x29 ;()) right parenthesis
char = 0x2A ;(*) asterisk
char = 0x2B ;(+) plus sign
char = 0x2C ;(,) comma
char = 0x2D ;(-) hyphen, minus sign
char = 0x2E ;(.) full stop
char = 0x2F ;(/) solidus
char = 0x3A ;(:) colon
char = 0x3B ;(;) semicolon
char = 0x3C ;(<) less-than sign
char = 0x3D ;(=) equals sign
char = 0x3E ;(>) greater-than sign
char = 0x3F ;(?)question mark
char = 0x40 ;(@) commercial at
char = 0x5B ;([) left square bracket
char = 0x5C ;(¥) reverse solidus
char = 0x5D ;(]) right square bracket
char = 0x5E ;(^) circumflex accent
char = 0x5F ;(_) low line
char = 0x60 ;(`) grave accent
char = 0x7B ;({) left curly bracket
char = 0x7C ;(|) vertical line
char = 0x7D ;(}) right curly bracket
char = 0x7E ;(~) tilde
```

```
char = 0x7F ;delete, or rubout
char = 0x80 ; undefined
char = 0x81 ; undefined
char = 0x82 ; undefined
char = 0x83 ; undefined
char = 0x84 ; undefined
char = 0x85 ; undefined
char = 0x86 ; undefined
char = 0x87 ; undefined
char = 0x88 ; undefined
char = 0x89 ; undefined
char = 0x8A ; undefined
char = 0x8B ; undefined
char = 0x8C ; undefined
char = 0x8D ; undefined
char = 0x8E ; undefined
char = 0x8F ; undefined
char = 0x90 ; undefined
char = 0x91 ; undefined
char = 0x92 ; undefined
char = 0x93 ; undefined
char = 0x94 ; undefined
char = 0x95 ; undefined
char = 0x96 ; undefined
char = 0x97 ; undefined
char = 0x98 ; undefined
char = 0x99 ; undefined
char = 0x9A ; undefined
char = 0x9B ; undefined
char = 0x9C ; undefined
char = 0x9D ; undefined
char = 0x9E ; undefined
char = 0x9F ; undefined
char = 0xA0 ;no-break space
char = 0xA1 ;inverted exclamation mark
char = 0xA2 ;cent sign
char = 0xA3 ;pound sign
char = 0xA4 ;currency sign
char = 0xA5 ;yen sign
char = 0xA6 ;broken bar
char = 0xA7 ;paragraph sign, section sign
char = 0xA8 ;diaeresis
char = 0xA9 ;copyright sign
char = 0xAA ;feminine ordinal indicator
char = 0xAB ;left angle quotation mark
char = 0xAC ;not sign
char = 0xAD ;soft hyphen
char = 0xAE ;registered trade mark sign
char = 0xAF ;macron
char = 0xB0 ;ring above or degree sign
char = 0xB1 ;plus/minus (+/-) sign
```

```
char = 0xB2 ;superscript 2
char = 0xB3 ;superscript 3
char = 0xB4 ;acute accent
char = 0xB5 ;micro sign
char = 0xB6 ;pilcrow or paragraph sign
char = 0xB7 ;middle dot
char = 0xB8 ;cedilla
char = 0xB9 ;superscript 1
char = 0xBA ;masculine ordinal indicator
char = 0xBB ;right angle quotation mark
char = 0xBC ;vulgar fraction one quarter
char = 0xBD ;vulgar fraction one half
char = 0xBE ;vulgar fraction three quarter
char = 0xBF ;inverted question mark
char = 0xD7 ;multiplication sign
char = 0xF7 ;division sign
;
; Digits
char = 0x30 ;(0) digit zero
char = 0x31 ;(1) digit one
char = 0x32 ;(2) digit two
char = 0x33 ;(3) digit three
char = 0x34 ;(4) digit four
char = 0x35 ;(5) digit five
char = 0x36 ;(6) digit six
char = 0x37 ;(7) digit seven
char = 0x38 ;(8) digit eight
char = 0x39 ;(9) digit nine
;
; Latin Alphabet
char = 0x41,0x61,0xC0,0xE0,0xC1,0xE1,0xC2,0xE2,0xC3,0xE3,0xC4,0xE4,0xC5,0xE5
;     A, a, A-grave, a-grave, A-acute, a-acute, A-circumflex,
;     a-circumflex, A-tilde, a-tilde, ;A-diaeresis, a-diaeresis,
;     A-ring, a-ring
char = 0x42, 0x62 ;letter B, b
char = 0x43, 0x63, 0xC7, 0xE7
;     letters C, c, C-cedilla, c-cedilla
char = 0x44, 0x64 ;letter D, d
char = 0x45, 0x65, 0xC8, 0xE8, 0xC9, 0xE9, 0xCA, 0xEA, 0xCB, 0xEB
;     E, e, E-grave, e-grave, E-acute, e-acute, E-circumflex,
;     e-circumflex, E-diaeresis, e-diaeresis
char = 0x46, 0x66 ;letter F, f
char = 0x47, 0x67 ;letter G, g
char = 0x48, 0x68 ;letter H, h
char = 0x49, 0x69, 0xCC, 0xEC, 0xCD, 0xED, 0xCE, 0xEE, 0xCF, 0xEF
;     I, i, I-grave, i-grave, I-acute, i-acute, I-circumflex,
;     i-circumflex, I-diaeresis, i-diaeresis
char = 0x4A, 0x6A ;letter J, j
char = 0x4B, 0x6B ;letter K, k
char = 0x4C, 0x6C ;letter L, l
char = 0x4D, 0x6D ;letter M, m
```

```
char = 0x4E, 0x6E, 0xD1, 0xF1
     ;letters N, n, N-tilde, n-tilde
char = 0x4F,0x6F,0xD2,0xF2,0xD3,0xF3,0xD4,0xF4,0xD5,0xF5,0xD6,0xF6,0xD8,0xF8
;     O, o, O-grave, o-grave, O-acute, o-acute, O-circumflex,
;     o-circumflex,O-tilde, o-tilde, O-diaeresis, o-diaeresis,
;     O-stroke, o-stroke
char = 0x50, 0x70 ;letter P, p
char = 0x51, 0x71 ;letter Q, q
char = 0x52, 0x72 ;letter R, r
char = 0x53, 0x73 ;letter S, s
char = 0x54, 0x74 ;letter T, t
char = 0x55,0x75,0xD9,0xF9,0xDA,0xFA,0xDB,0xFB,0xDC,0xFC
;     U, u, U-grave, u-grave, U-acute, u-acute,
;     U-circumflex, u-circumflex, U-diaeresis, u-diaeresis
char = 0x56, 0x76 ;letter V, v
char = 0x57, 0x77 ;letter W, w
char = 0x58, 0x78 ;letter X, x
char = 0x59, 0x79, 0xDD, 0xFD, 0xFF
;     letters Y, y, Y-acute, y-acute, y-diaeresis
char = 0x5A, 0x7A ;letter Z, z
;
; Alpha characters not used in English, French or German:
char = 0xD0, 0xF0 ;icelandic capital letter Eth, small letter eth
char = 0xDE, 0xFE ;icelandic capital letter Thorn, small letter thorn
```

付 録　A

国際化に関連するディレクトリとファイル

この章では、国際化とローカライゼーションに関係のある Open Client/Server のディレクトリとファイルについて説明します。

この章の内容は、次のとおりです。

概要

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取り出します。Sybase リリース・ディレクトリの次の 3 つのディレクトリにローカライゼーション情報が入っています。

- *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。
- *charsets* ディレクトリには、サポートする各文字セットの変換ファイルと照合順ファイルが入っています。
- UNIX の *config* ディレクトリと Microsoft Windows の *ini* 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が入っています。
- *collate* ディレクトリは、Adaptive Server Enterprise が並べ替えに使用します。各文字セットには、Adaptive Server Enterprise がデータの並べ替えに使用するソート順が 1 つ以上定義されています。

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をサポートするファイルが含まれています。これらのファイルは、インストール中に Sybase リリース・ディレクトリ構造の正しいロケーションにロードされます。

注意 インストール処理では、コネクティビティ用の Open Client/Server 言語モジュールは Sybase リリース・ディレクトリの正しいロケーションに自動的にロードされます。

locales ディレクトリ

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ています。

- ロケール名を言語、文字セット、照合順にマップするロケール・ファイル (*locales.dat*)。
- Open Client/Server 製品用のローカライズされたエラー・メッセージが入っている *message* ディレクトリ。
- 以前のリリースの Open Client/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。

ロケール・ファイル

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

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

「第 5 章 ロケール・ファイルの編集」を参照してください。

ローカライズされたメッセージ・ファイル

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) を使用して、Open Client/Server アプリケーションはさまざまな言語でメッセージを生成できます。

すべての Open Client/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。

language_name サブディレクトリ

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

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

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

Unicode ディレクトリ

すべての Open Client/Server メッセージ・ファイルは、Unicode UTF-8 文字セットを使用し、必要に応じて UTF-8 をその他の文字セットに変換します。

charsets ディレクトリ

charsets ディレクトリは次のディレクトリとファイルから構成されています。

- それぞれの文字セット用の *charset_name* サブディレクトリ。それぞれの *charset_name* サブディレクトリには、サポートする各文字セットの照合順ファイルが入っています。
- *unicode* ディレクトリには、Unilib によって使用される Unicode 変換ファイルが入っています。

照合順ファイル

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*

これらのファイルでは十分でない場合は、カスタマイズした照合順ファイルを作成できます。この実行方法の詳細については、「カスタム照合順ファイルの作成」(58 ページ) を参照してください。

照合順はロケール・ファイルのエントリで指定されます。ロケール・ファイルのエントリで照合順が指定されていない場合、ロケールについてはバイナリ・ソート順が使用されます。「第 6 章 照合順の作成または変更」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 標準と同等) 文字セットの変換設定情報が入っています。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 ディレクトリと *ini* ディレクトリ

UNIX の *config* ディレクトリと Microsoft Windows の *ini* ディレクトリには、グローバル識別子ファイル (*objectid.dat*) が入っています。

グローバル・オブジェクト識別子ファイル

グローバル・オブジェクト識別子ファイル *objectid.dat* は、オブジェクトに使用される可能性のあるすべてのローカル名とユニークなグローバル・オブジェクト識別子を対応させます。

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。

オブジェクト識別子ファイル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・エントリの形式は次のとおりです。

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

ここで、

- *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。

オブジェクト識別子ファイルの例

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]
     1.3.6.1.4.1.897.4.6.6 = csfkrb5
```

オブジェクト識別子ファイルの編集

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 B

外部ローカライゼーション・ファイルの構文

この章では、外部ローカライゼーション・ファイルの構文について説明するとともにサンプル・ファイルも示します。ロケール・ファイル (locales.dat) や照合順ファイル (*sort_order_name.srt*) などの外部ローカライゼーション・ファイルを作成したり更新したりする場合にこの情報を使用してください。

この章の内容は、次のとおりです。

ローカライゼーション・ファイルの構文規則

すべての外部ローカライゼーション・ファイルは、次のような基本的な構文規則に従います。

- 「コメント」はコメント文字で始まり、その行の終わりまで続きます。ファイルの最初の行の先頭の文字は、そのファイルのコメント文字として定義されます。
- 「セクション」はセクション見出しで始まり、セクションにはエントリが指定されています。セクション見出しでは左右のデリミタが使用されます。セクション見出しの長さの最大値は、左右のデリミタも含めて 63 バイトです。

  ファイル内の最初の行がコメント文字で始まっていない場合は、そのファイルのセクション見出しのデリミタを定義します。その先頭の文字は左デリミタとして定義され、最後の文字は右デリミタとして定義されます。
- 「エントリ」の形式は次のようになります。

  keyword = *value_list*

  ここで、

  - *keyword* はエントリ・キーワードであり、長さの最大値は 63 バイトです。

- *value_list* は、リスト・セパレータ文字で区切られた 1 つまたは複数の値のリストです。それぞれの値には引用符付きの文字列、引用符なしの文字列、または 16 進数のうちのどれを指定してもかまいません。*value_list* が指定されない場合、エントリ・キーワードの値として長さ 0 の文字列 (つまり、NULL ターミネータのみから構成される文字列) が割り当てられます。

  最後の行を除くそれぞれの行の最後にエスケープ文字を指定すると、*value_list* を複数の行に渡って設定できます。

  value_list の長さの最大値は 511 バイトです。

1 つの行に指定できるエントリは 1 つだけです。エントリの前にはタブとスペースを指定できます。

- 「値」には 16 進数、引用符付きの文字列、または引用符なしの文字列のうちのどれを指定してもかまいません。
  - “0x” で始まる引用符なしの文字列は、16 進数として解釈されます。
  - リスト・セパレータやスペースが含まれていない文字列の場合は、引用符を付ける必要はありません。引用符付きの文字列の中にリスト・セパレータとスペースが含まれている場合は、その前にエスケープ文字があるものとして処理されます。
  - 文字列を囲むのにアポストロフィまたは引用符を使用することもできます。アポストロフィ (‘) は、引用符で囲まれた文字列 (“string”) の中で使用できます。また引用符は、アポストロフィで囲まれた文字列の中で使用できます。

    アポストロフィと引用符のどちらかが繰り返される場合、この 2 つの文字は文字列のデリミタとしてではなく、本来のその文字 1 文字として処理されます。たとえば、“Jean's book” などがその例です。

ローカライゼーション・ファイルのセクション

さまざまなファイルには異なるタイプのセクションがあり、さまざまなタイプのセクションには異なるエントリ・キーワードがあります。

この項では、すべてのローカライゼーション・ファイルに共通しているセクションについて説明します。

表 B-1 は、特定のファイルに固有であるセクションについて説明してある項を示します。

表 *B-1:* ファイルに固有のセクションについての参照箇所

ファイル名	参照箇所
ロケール・ファイル (*locales.dat*)	「ロケール・ファイルのセクションとエントリ」 (46 ページ)
照合順ファイル (*sort_order_name.srt*)	「照合順ファイルのセクションとエントリ」 (56 ページ)

表 B-2 は、すべての外部ローカライゼーション・ファイルに共通しているセクションについて説明します。

表 *B-2:* ローカライゼーション・ファイル内の標準セクション

セクション	説明	例

ファイル・フォーマット・セクション	このセクションはオプションである。
	このセクションを使用する場合、その形式は次のようになる。
	`[file format]`
	`version = version_number`
	`list_separator = list_separator_char`
	`escape = escape_char`
	ここで、
	• *version_number* はバージョン番号。
	• *list_separator_char* はそのファイルに使用するリスト・セパレータ文字。
	• *escape_char* はそのファイルに使用するエスケープ文字。このセクションを指定しない場合は、“list_separato” はデフォルトで “,” (カンマ)、“escape” はデフォルトで “¥” (円記号) に設定される。
		`[file format]`
		`version = 1`
		`list_separator =,`
		`escape = ¥`

版権セクション	このセクションはオプションである。
	このセクションを使用する場合、その形式は次のようになる。
	`[copyright]`
	`copyright = "copyright_statement"`
	ここで、*copyright_statement* は文字列。
		`[copyright]`
		`copyright = "Copyright¥ Excellent Products, Inc."`

ローカライゼーション・ファイルの例

この項で示す照合順ファイルの部分的な例では、「ローカライゼーション・ファイルの構文規則」(75 ページ) で説明した構文規則のいくつかが示されています。

このファイルを参照する場合には、次の点に注意してください。

- 最初の行では、コメント文字をセミコロンと定義しています。セミコロンで始まる後続の行またはフレーズはコメントになります。
- 2 番目の行 [sortorder] は、ソート順セクションの見出しです。このセクションに指定されているエントリでは照合順についての記述と定義を行っています。このファイル例では、オプションである版権セクションとファイル・フォーマット・セクションは指定されていません。
- このファイルのリスト・セパレータはカンマ (デフォルト) です。
- このファイルのエスケープ文字は円記号 (デフォルト) です。
- スペースが含まれている値は、“description =” の値のように引用符で囲みます。

注意 省略記号 “...” は、実際のファイルの内容が省略されていることを意味します。

```
; semi-colon is the comment character
[sortorder]
;-----------------------------------------------------------------
; Overview
; --------
; Case-sensitive sort order based on the ISO 8859-1 code set.
; Uppercase characters sort before lowercase counterparts.
;
; Ligatures and sort doubles
; --------------------------
; AE, ae ligatures
; German sharp-s ligature with “ss”
;
; Sort order
; ----------
; 1. non-alphanumeric characters in binary order
; 2. numeric digits
; 3. alphabetic characters used in English, French, German
; 4. Alphabetic characters not used in English, French, German
;
; Format
; ------
; Default formatting values.There is no [file format] section.
;-----------------------------------------------------------------class = 0x01
id = 0x33
```

```
menuname = “Case-sensitive dictionary sort order”
name = dictionary
charset = iso_1
description = “Dictionary sort order for use with English,¥ French and German.
ISO 8859-1,case sensitive.”
;
; Ligatures for English, French, German
lig = 0xC6, after AE
lig = 0xE6, after ae
lig = 0xDF, after ss
;
; Control characters
char = 0x01(SOH) start of heading
...
char = 0x1F;(US) unit separator
;
; All non-alphanumeric characters, including punctuation,sorted
; by numerical ordering
char = 0x20;( ) space
...
char = 0xF7;division sign
;
; Digits
char = 0x30;(0) digit zero
...
char = 0x39;(9) digit nine
;
; Latin alphabet
char = 0x41,0x61,0xC0,0xE0,0xC1,0xE1,0xC2,0xE2,0xC3,0xE3,0xC4,
0xE4,0xC5,0xE5
; letter A, a, A-grave, a-grave, A-acute, a-acute, A-circumflex,
; a-circumflex, A-tilde, a-tilde, A-diaeresis, a-diaeresis,
; A-ring, a-ring
...
char = 0x5A,0x7A;letter Z,z
;
; Alphabetic characters not used in English, French, German
char = 0xD0,0xF0;Icelandic letter Eth, eth
...
```

用語解説

UTF-16

UTF-16 コード化は、16 ビット形式の UCS 変形フォーマットです。UTF-16 では、各 UCS-2 コード値はそれ自体を表し、現在定義されている文字は、すべて 2 バイト長です。BMP (Basic Multilingual Plane: 0..0xFFFF) の範囲外のコード値は、サロゲート・ペアという特別なコードのペアを使用して表現します。

UTF-8

UTF-8 コード化は、8 ビット形式の UCS 変形フォーマットです。最大 4 バイトまでの長さの文字が使用されます。

Unicode

Unicode の標準で定義される汎用の 16 ビット・コード化文字セット。Unicode のバージョン 1.1 は、国際標準の汎用文字セット ISO 10646 とコードごとにすべて一致しています。

大文字と小文字を区別する (case-sensitive) ●以下の各用語解説をあいうえお順にソートしてください。●

照合順について使用される場合、その照合順では大文字と小文字が区別されることを意味します。

グリフ (glyph)

文字のグラフィック表現。たとえば、文字 “f” はグリフの “f” または “*f*” で表すことができます。

コード化文字セット変換 (coded character set conversion)

文字のコード化をある数値コードの集合から別の数値コードの集合に変えること。

クライアントとサーバが異なる文字セットを使用する場合、コード化文字セット変換を行うことによってデータを同じように解釈します。

コード化文字セット (coded character set)

それぞれの文字に数値コードが割り当てられている文字セット。「コード・ページ」とも呼ばれます。

コード化 (encoding)

文字セットのコード化の場合は、数値コードを使用してそれぞれの文字をユニークに識別することを意味します。

国際化 (internationalization)

アプリケーションの複数言語、文化的な規則への対応を可能にするプロセス。国際化されたアプリケーションは、そのアプリケーションが稼働している地域に適した言語と文化的慣例を使用します。

合字 (ligature)

複数の文字としてソートされる単一の文字。たとえば “→” は “AE” としてソートされ、“β” は “ss” としてソートされます。

照合順 (collating sequence)

システムがテキストをソートする順序。

シングルバイト文字セット (single-byte character set)

すべての文字がシングルバイトを使用してコード化される文字セット。

ソート・ダブル (sort double)
照合順で単一の文字としてソートされる一対の文字。たとえばスペイン語の “ch” など。

ソート順 (sort order)
「照合順」の説明を参照してください。

二重音字 (digraph)
「合字」の説明を参照してください。

表意文字 (ideograph)
中国語と日本語の書き言葉で使用されるような、ある意味を表す文字または記号。

マルチバイト文字セット (multibyte character set)
EUC JIS とシフト JIS など、2 バイト以上を使用してコード化される文字を含む文字セット。マルチバイト文字セットには、可変長の文字が含まれる場合もあります。

文字セット (character set)
文字、表意文字、数字、記号、制御コードを含む文字とグリフの有限集合。「シングルバイト文字セット」と「マルチバイト文字セット」の説明も参照してください。

文字 (character)
“e”、“ë”、“5”、または **“¿”** など、ネイティブ言語でデータを表す要素の集合の中の 1 つ。

ローカライゼーション (localization)
アプリケーションが特定の言語とその言語に関連する文化的慣例を使用して稼働するように設定する処理。

ロケール・ファイル (locales file)
言語、文字セット、照合順にロケール名をマップする Sybase 独自のファイル。Open Client/Server 製品は、ローカライゼーション情報をロードするときにこのロケール・ファイルを調べます。

ロケール構造体 (CS_LOCALE) (locales structure)
Client-Library アプリケーションと Server-Library アプリケーションでカスタム・ローカライゼーション値を定義するのに使用される CS-Library 構造体。CS-Library ルーチン cs_loc_alloc と cs_loc_drop は、ロケール構造体の割り付けと削除を行います。CS-Library ルーチン cs_locale は、ロケール構造体を情報とともにロードします。

ロケール (locale)
1. 特定の地理または国語の地域。2. 特定の地理または国語の地域に関係のある情報の集まり。

索引

O

P

S

U

あ

え

お

か

き

け

こ

し

す

せ

そ

て

に

は

ひ

ふ

へ

め

も

ゆ

れ

ろ