Global Deployment Guide

Version 7.8, Rev. A
November 2005

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404

Contents

Chapter 4: Configuring Global Deployments

Chapter 5: Deploying with Global Time Zone

Chapter 6: Localizing Global Deployments

Index

1 What's New in This Release

What's New in Global Deployment Guide, Version 7.8, Rev. A

Table 1 lists changes described in this version of the documentation to support release 7.8 of the software.

Table 1. New Product Features in Global Deployment Guide, Version 7.8, Rev. A

Topic	Description
"Global Deployment Terminology" on page 9	Updated several definitions.
"About the Database Collation Sequence" on page 23	Updated topic.
"Scenarios for Global Deployment"	Deleted unnecessary section from Chapter 3, "Planning Global Deployments."
"About Parameters for Language and Locale" on page 39	Updated and expanded the descriptions of Language Code and OM - Resource Language Code parameters. In particular, the user interface language is never determined by the OM - Resource Language Code parameter.
"Configuring Language Support on Windows ME/98" on page 47	Deleted references to unsupported platforms (Windows 95 and Windows NT 4.0).
"Configuring Language Support on IBM OS/2 and Sun Solaris 8"	Deleted obsolete section, which referenced unsupported platforms, from Chapter 4, "Configuring Global Deployments."
"Configuring the Calendar" on page 54	Added topic.
Chapter 5, "Deploying with Global Time Zone"	The UTC-related topics are now located in a separate chapter (formerly, they were in Chapter 4, "Configuring Global Deployments"). Updated, reorganized, and expanded the UTC-related topics.
"Localizing Lists of Values and Multilingual Lists of Values" on page 84	Added topic.
"Localizing an Unshipped Language" on page 85	Updated topic.

What's New in Global Deployment Guide, Version 7.8

Table 2 lists changes described in this version of the documentation to support release 7.8 of the software.

Table 2. New Product Features in Global Deployment Guide, Version 7.8

Topic	Description
Chapter 5, "Deploying with Global Time Zone"	Configuring the global time zone feature (based on Universal Time Coordinated, or UTC) no longer requires the database server machine to be set to a GMT time zone. **NOTE:** For Rev. A, the UTC-related topics are now located in Chapter 5, "Deploying with Global Time Zone."

2 Overview of Global Deployments

This chapter provides overview information about global deployments of Siebel Business Applications. It includes the following topics:

Global Deployment Terminology

This guide uses many specialized terms. Table 3 on page 10 identifies and provides brief descriptions for some of these. Expanded discussions of some of these terms and related concepts are provided in topics following the table.

Table 3. Global Deployment Terminology

Term	Meaning
Character Set	A group of characters (alphanumeric, text, or special) usually associated with one or more languages or scripts. There are many character sets used in the computing industry. Character sets are identified by a character set name, such as Western European or Latin 1. These names are not well-standardized and many character sets have multiple names. However, you can use formal identifiers to clearly specify a character set when necessary. See also "About Supported Character Sets" on page 19.
Character Set Encoding	Also known as character encoding. A specific computer representation of a character set. Some character sets can have multiple encodings—for example, Western European or Latin 1 is encoded differently in ISO 8859-1, 8859-15, ANSI 1252 (Microsoft standard), and EBCDIC. Unicode also comes in different encodings, such as UTF-8 or UTF-16. In general, the differences in encodings are between ISO, ANSI, and EBCDIC. Aside from Unicode, the most prominent example of a character set with multiple encodings is JIS (Japan Industrial Standard). Shift-JIS, EUC, and ISO 2022-JP are all encodings based on JIS. As with character set names, industry standardization is minimal and there are multiple names for the same encoding. The character set encoding is also known as a code page or codepage (one word), which often refers to a vendor implementation of a character set encoding. For Microsoft Windows, the term code page is used for ANSI code page (Windows) and OEM code page (DOS), but not for ISO character sets. IBM uses a numbering system which is similar, and often identical, to Microsoft. However, IBM renumbers extensions while Microsoft does not, which can lead to references such as "Use IBM 943 with Siebel applications and MS 932; IBM 932 is an older version." See also "About Character Set Encodings and Siebel Applications" on page 21.
Code Point	A single data value representing a single character in a code page.
Global Deployment	The process of installing, configuring, testing, and deploying Siebel Business Applications in more than one locale and language.
Internationalization	The process of making a software product that can correctly process data for any customer, including data entry and display, through proper locale. For more information, see "Internationalization" on page 15.

Table 3. Global Deployment Terminology

Term	Meaning
Internet Corporation for Assigned Names and Numbers (ICANN) Internet Assigned Numbers Authority (IANA)	Internet bodies that manage, or have managed, Internet addresses, domain names and protocol parameters. IANA was replaced by ICANN, which was formed in 1998. IANA or ICANN conventions are used for the World Wide Web, email, and XML.
Language	The language or languages of the Siebel Business Applications software installed on the system. Language Pack is another term for languages you install with Siebel software or the Siebel Database, or add to existing installations. For more information, see "Language" on page 13.
Locale	A set of user preference information related to the user's language and cultural conventions, including the formatting or presentation style of data such as dates, time, numbers, and currency. For more information, see "Locale" on page 14 and "About Date, Time, Number, and Currency Formatting" on page 18.
Localization	The process of readying a product for use in a particular target country. For more information, see "Localization" on page 16.
Multicurrency Support	A feature that allows automatic currency conversion and currency formatting. For more information, see "About Date, Time, Number, and Currency Formatting" on page 18.
Non-Unicode (Traditional) Character Set	Non-Unicode (traditional) character sets refer to character sets other than Unicode and imply that the character set supports a restricted set of characters. Typically, a traditional character set supports the alphabet of a single language or of a collection of languages that use the same or similar alphabets. See also "About Supported Character Sets" on page 19.
Platform	A platform includes the operating system of the various entities of a Siebel deployment; the database, the Siebel Servers, and the clients and the character set used by these entities. For more information, see "Platform" on page 12.

Table 3. Global Deployment Terminology

Term	Meaning
Script	A system of writing that requires graphical symbols to be placed in a certain order to communicate information. The symbols can be based on an alphabet, or on pictures of objects in the world around us, or on some other system. The Roman script (sometimes called the Roman alphabet) is a script that uses 26 symbols to represent sounds made by the human mouth, organized into an alphabet. Originally the script used to write Latin (the language of the Romans), it has been extended with diacritics on many of the characters to express other sounds present in Western European languages. (In a Siebel development context, a very different meaning of the term script is a program written using a language such as Siebel eScript.)
Unicode Character Set	A character set defined by the Unicode Consortium that is the union of most of the major character sets used in the computing industry. See also “About Supported Character Sets” on page 19.
Universal Time Coordinated (UTC)	Also known as coordinated universal time. A time scale defined and recommended by the International Radio Consultative Committee (CCIR), and maintained by the Bureau International des Poids et Mesures (BIPM). The global time zone feature uses UTC. For more information, see Chapter 5, “Deploying with Global Time Zone.”

Platform

The *platform* determines what data can be processed in a Siebel deployment. The character set encoding and operating system language of a platform will determine what data can be handled correctly and what data will not be handled correctly by the platform.

This document uses the term platform in several places to discuss deployment options as well as specific functionality available in Siebel Business Applications.

Siebel Business Applications generally do not support mixed character encoding environments. The reason is that it is not technically possible to manage an environment that uses multiple character set encodings on databases and servers without a genuine risk of losing data in the process.

For example, suppose a database is set up with a Western European character set encoding and a user tries to insert Japanese data through a Siebel Server set up for Japanese. Depending on the actual database, the effect could be that the user’s data would be rejected and not stored in the database, or the data could get converted by the database and stored incorrectly as unreadable characters (substitution characters), resulting in loss of the original Japanese data.

In some cases a user may try to enter data into the application and receive an error message saying that the language of the text entered “is not compatible with the database language” or that the length of the text entered “is bigger than the corresponding length allocated in the database.”

The above error may occur when the character set of the data does not match the character set of the database, and cannot be converted without data loss. Or, the error may occur when the data string is too long to fit into the database column.

Verify that your database is set up for Unicode, if you are using Unicode. Also check that all extension columns are large enough to accept characters larger than one byte in length. For example, in UTF-8 (supported on Oracle databases), accented characters use two bytes, so extension columns must be sized to be able to include such characters.

NOTE: If your database uses Code Page 932 (or 936 on DB2, or JA16SJIS on Oracle), it is strongly recommended to set the parameter UseANSIControlsForCP to CP932 in the [SWE] section of the configuration file (such as uagent.cfg for Siebel Call Center). If you do not set the parameter in this way, users may sometimes be able to enter more text in a field than will be saved in the database. This parameter applies only for high-interactivity clients. It is assumed that the client machine uses the same code page as the database.

Language

The *language* for a Siebel application can mean multiple things, and may involve different system or application elements. These elements are independent from the language of the data that the user enters in the Siebel Database. You must install seed data according to how you want to use languages in the Siebel applications.

- The *primary language* (sometimes called the *base language*) is the first language installed for this Siebel installation.
- The *active language* is the language in effect for an individual user's session and the language of user interface elements, including multilingual lists of values (MLOVs). This language is also used for system messages (if the resource language is not separately defined).

  For a Siebel Web Client session, the language is determined by the Application Object Manager (AOM) component invoked, and cannot be changed by the user—except by logging into a different language-specific AOM. For the Mobile or Developer Web Client, the user can explicitly specify the language to use for a given session.
- The *resource language*, if it is defined, is used as the default language for system messages.

Siebel Language Packs install the language-specific run-time environment on the Siebel Server: repository (SRF) files, DLLs, configuration (CFG) files, reports, error messages, help files, and so on. Installing languages on the Siebel Database loads language-specific seed data.

The languages allowed in data are constrained only by the character encoding of the database platform. For example, although a user may be using a U.S. English version of a Siebel application installed with a Western European code page database, the user can enter or view contact data in French, because all French characters are representable in the Western European code page.

With a Unicode code page, and appropriate fonts locally installed, languages using dissimilar character sets, such as French and Japanese, may be used together.

Language codes used by Siebel applications use a three-letter code, such as ENU for U.S. English, FRA for French, THA for Thai, and so on. Using language codes with only two characters does not work and is not supported.

See also "About Parameters for Language and Locale" on page 39 and "Creating Language and Locale Records" on page 86.

Locale

A *locale* is based on the language, country (territory), and the character set. Siebel applications cannot control the character set supported by the database and do not have the concept of a country, so the locale is primarily based on the language. However, locales are defined in the Siebel seed data and can be associated with Application Object Manager components.

The locale includes a collection of user profile inputs, including keyboard layout and the formats used for numbers, dates, currencies, and times.

The Siebel Web Client adopts the locale settings in effect for the Application Object Manager component on the Siebel Server.

The Siebel Mobile Web Client and Developer Web Client adopt the locale settings defined in the client operating system's regional settings.

For more information about locales defined in Siebel applications, see the *Applications Administration Guide*.

Different types of locales are described below:

- **User locale.** The current language and country settings active for this session.

  You can set a locale to provide data to users in their native format, including the formatting of numeric information such as numbers, times, dates, and currencies. Typically, user locales contain the symbols for the thousand separator, decimal point, negative number representation, time separator, short data format, long data format, and currency symbols. A country specification is often used to select default values for user locale settings.

  Both the Siebel Database and the Siebel application have locale settings, which are independent of the operating system (except for the Siebel Mobile and Developer Web Client).

- **Input locale.** The current language used for entering data from the keyboard.

  The input locale affects the layout of keys on the keyboard, and for some languages, the way in which those key entries are then processed before entering the data into the current form on the screen. It is used to describe the language being entered and the input method, which could be a particular keyboard layout or a speech-to-text converter.

  Keyboard layout is a defined input locale that correlates the keys on the keyboard to their subsequent character definition mapping within the code page of the operating system.

- **System locale.** If you are using a Microsoft Windows operating system, this is a systemwide setting that designates which code page is used as the default for all users on the system. If you are using a UNIX operating system, the settings for formatting and code page locales are *not* systemwide. These code pages and fonts allow non-Unicode applications to run as they would on a system localized to the language of the system locale.

  For more information about specifying the system locale on UNIX, see the *Siebel Installation Guide for UNIX: Servers, Mobile Web Clients, Tools*.

  NOTE: If you are using a Windows operating system, you *must* restart the system after changing a system locale.

Locale Usage

You can use locale rules to vary the appearance of data for different regions of your implementation. Typically, this data would include dates and times, numbers, and currencies.

For example, the date and time *thirty minutes past four in the afternoon on May nine, year two thousand-and-five* can appear differently depending on the locale. It may appear as:

- 05/09/05, 04:30 PM, if the locale used is English American.
- 09.05.2005, 16:30, if the locale used is German.

Locales specify thousand separators and decimal symbols for numbers. They determine the position of the currency symbol in relation to the currency amount.

Locales also guide what characters are available through the computer keyboard. Users can remap their keyboards through the locale setting to get access to additional characters when typing.

Internationalization

Internationalization includes designing software to handle and display data, such as text, diagrams, and numbers, according to the orthography or rules of the language as used in a particular locale. Internationalization is often abbreviated as I18N, because there are 18 characters between the initial I and the terminal N.

The software might have to input, display, and print characters, sort text, and recognize numbers and dates in different formats, and display and print text right-to-left as well as left-to right. Therefore, certain engineering features must be incorporated into the code to handle these requirements.

Developing an internationalized program means that the feature and code designs do not make assumptions based on a single language or locale and that the source code base simplifies the creation of different language editions of a program.

Some aspects of internationalization include:

- A base version enabled for international environments
- Localizable items separated from the core functionality on which they are running

- Software that takes advantage of supporting platforms, such as the Windows operating systems and the database platform the software is running on

Your Siebel application has been internationalized. Specific features include:

- A base version, enabled for international environments
- Support for localization built into the data model
- Support for separate language-specific modules (where necessary)

  For example, some DLLs are language-independent, while other DLLs are language-dependent. Language-dependent DLLs are located in the language-dependent installation directory that they support.
- Euro (€) currency support and EMU triangulation
- String, number, and date handling
- Support for multilingual user data, such as:
  - Multilingual picklists (MLOV seed data)
  - Multilingual data for product- and catalog-related entities
- Support for major Unicode and non-Unicode (traditional) character sets

  For more information, see *System Requirements and Supported Platforms* on Siebel SupportWeb.
- The ability to support both left-to-right and right-to-left displays, referred to as *bidirectionality*

Localization

Localization is the process of readying a product for use in a particular target country. Localization is often abbreviated as L10N, because there are 10 characters between the initial L and the terminal N. (The product must have been internationalized or else most localization cannot be performed.)

Localization tasks are described in Chapter 6, “Localizing Global Deployments.”

Localization consists of two activities:

- **Translation.** Taking all applicable strings that appear on-screen in the application user interface and translating them into the language used in the target country.
- **Adaptation.** The process of making sure the product is suitable for use in the target country. Example activities are:
  - Modifying the user interface to display language-specific elements—for example, hiding or displaying fields or modifying the position, height, and width of controls to accommodate the target language. For example, if a target country does not have a governmental equivalent to a state, then the State field might be hidden for the target country.
  - Modifying images used in the application to those appropriate for the target country.

- Ensuring that the default configuration for the target country includes the right date format, currency, address format, salutations, names of provinces or states, and so on. User interface labels and master data may need to be modified.

  For example, a U.S.-specific term like *SSN* (Social Security number) is not translatable, but may be replaced with an equivalent term for the target country, such as *national ID number*.

  For another example, the State field is prepopulated with the names of the U.S. states. These values are incorrect in other countries that have states (or equivalent), such as Mexico and Brazil. Where applicable, the LOV containing state names should be replaced with the list of states (or equivalent) for the target country.

  Addresses use a single format for each language, and there are more than 400 address applets across the applications. For each supported language, Siebel Systems predefines the address formats for the target country. For example, the address format for France is used with the French language pack. French-speaking users in Canada will find that this is the wrong address format, so it should be changed. Similarly, the U.S. address format, used for the ENU language pack, is incorrect for English-speaking users outside of the U.S.

- Changing from a left-to-right display to a right-to-left display. (The ability to support both left-to-right and right-to-left displays, referred to as *bidirectionality*, is an internationalization feature.)

- Defining and implementing access control mechanism as appropriate for the users in the target country and the data they will be working with. Data may need to be visible in multiple countries or visible only in particular countries.

Siebel Business Applications are localized as required by the Siebel customer base. Local language releases are translated and elements of the user interface, including buttons, error messages, reports, online help, and log files, are configured to meet local requirements. An example for Spanish is shown in Figure 1 on page 18.

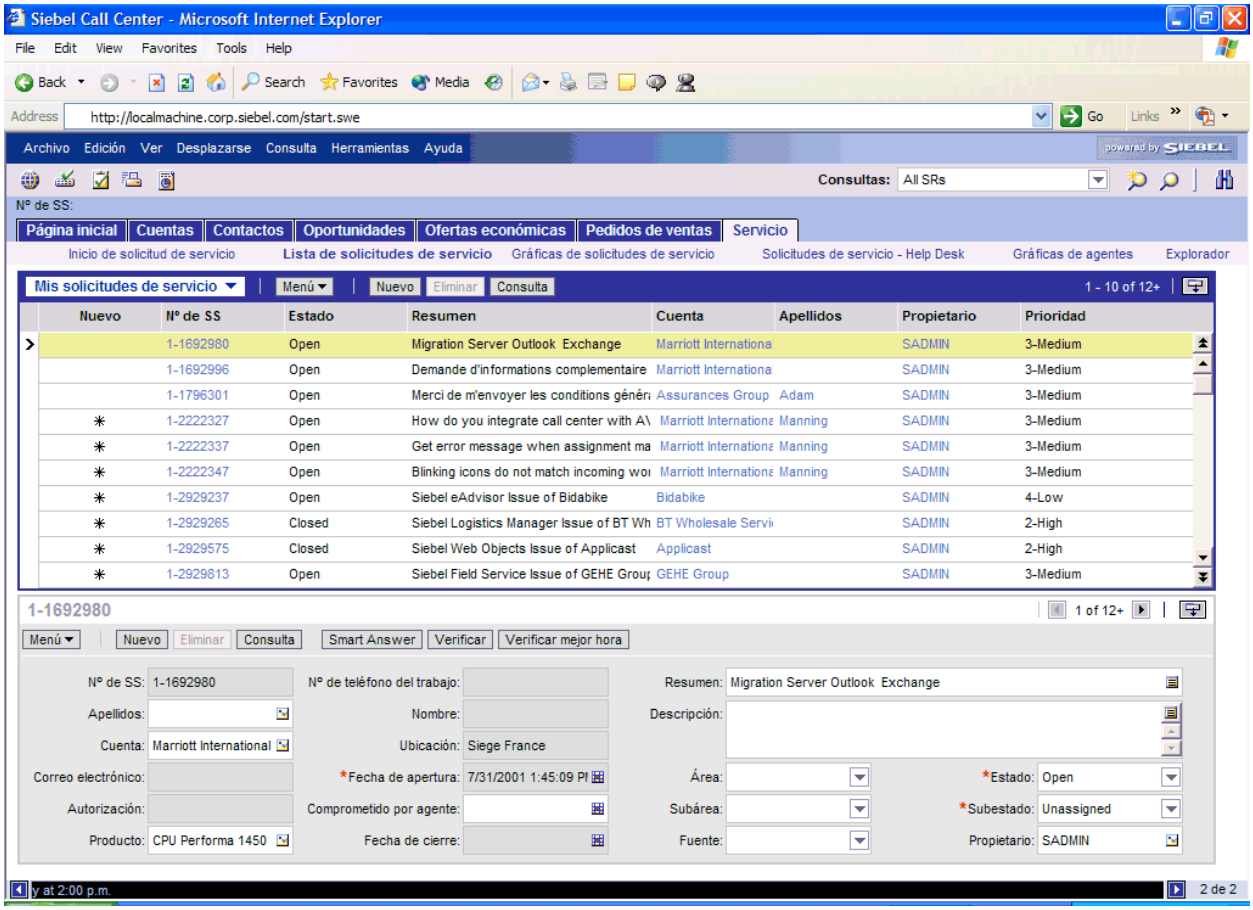

Figure 1. Example of Localized User Interface

The features that make the product internationalized are part of the software architecture; they do not require a special version of the product.

Customers must perform certain tasks to complete localization. The necessary tasks may vary according to the language requirements. For example, implementing any language that displays using a right-to-left directionality, such as Arabic or Hebrew, requires a particular set of tasks.

About Date, Time, Number, and Currency Formatting

Siebel Business Applications support formatting of data such as dates and time, numbers, phone numbers, and currency, based on locale settings. Some examples of differing formats based on locales include:

- Date and time
  - 03/10/04 (U.S. format, as *mm/dd/yy*)

  - 10.03.04 (German format, as *dd.mm.yy*)
- Number
  - 1,234.34 (U.S. format, with a comma as the digit grouping symbol and a period for the decimal symbol)
  - 1 234,34 (French format, with a space as the digit grouping symbol and a comma for the decimal symbol)
  - 1.234,34 (German format, with a period as the digit grouping symbol and a comma for the decimal symbol)
- Phone number
  - +33 1-23 42 34 56 (French phone number, as shown in U.S. regional settings)
  - (415) 295-5000 (U.S. phone number, as shown in U.S. regional settings)
- Currency
  - $32.45 (U.S. format, with U.S. dollar currency symbol in front of the amount)
  - 99.40 kr (Swedish format, with Krona currency symbol behind the amount)

The handling by software of multicurrency transactions for multinational businesses includes automatic currency conversion with full euro support. Siebel applications allow you to conduct currency transactions using multiple currencies, define additional currencies as needed. Currencies are converted as needed within the Siebel application, such as when rolling up forecasts.

For information about administering currency conversion, see the *Applications Administration Guide*. For information about configuring dual-currency display, see *Configuring Siebel Business Applications*.

About Supported Character Sets

This section provides information about Siebel-supported non-Unicode (traditional) and Unicode character sets. It includes the following topics:

- "Non-Unicode (Traditional) Character Sets" on page 19
- "Unicode Character Sets" on page 20

In this section, the terms character set and code page are used to cover closely related concepts used by the various platform vendors.

NOTE: Siebel Business Applications do not support any character that has been added to a font by mapping it to an open code point.

Non-Unicode (Traditional) Character Sets

Before the emergence of Unicode, non-Unicode (traditional) character sets were available to address storage and processing requirements for a specific language or group of languages.

Examples of non-Unicode character sets are Code Page 1252 for languages spoken in Western European countries as well as in the Americas and elsewhere, and Code Page 932 for the Japanese language.

Because of the regional aspect of non-Unicode character sets, character data for languages not part of the character set cannot be processed in the same environment. Therefore, when a need to process data belonging to multiple character sets arise, customers are forced to provide multiple environments.

Also, because character sets are expressed in code pages, the numeric representation of a character in one code page may be different from the representation in another code page, and often the character does not even exist.

For example, the letter a-umlaut (ä) in the Western European character set does not exist in the Arabic character set. In a Western European code page, such as 1252 or ISO 8859-1, the a-umlaut occupies code point E4 (Hex value). In an Arabic code page, such as 1256 or ISO 8859-6, the E4 code point is an Arabic character and not the a-umlaut. Thus, you cannot represent the a-umlaut character on an Arabic system, or represent the Arabic character in a Western European system.

There is a set of characters that are common in most generally used non-Unicode character sets and code pages. These characters are known as the ASCII characters. They include the common characters used in the English language and they occupy the first 128 code points (Hex 00-9F) in the non-Unicode code pages.

NOTE: It is the customer's responsibility to choose a character set that includes the characters required by the customer's business. Since the character set is a property of database configuration performed by the customer, Siebel Systems has no control over this setting. Choosing an inappropriate character set may require database reconfiguration later, and a corresponding need to convert large amounts of transaction data that has built up in the wrong character set. This is generally a time-consuming and costly experience. Character set conversion to Unicode must be done with the assistance of Siebel Expert Services.

For more information, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

Unicode Character Sets

To meet the needs of global operations, a number of software and hardware providers started the Unicode Consortium and created a Unicode standard during the 1990s. The repertoire of this international character code for information processing includes characters for the major scripts of the world, as well as technical symbols in common use. Unicode can represent 64 thousand planes of 64 thousand characters each. Unicode character encoding treats alphabetic characters, ideographic characters, such as Kanji, and symbols identically, which means that they can be used in any mixture with equal facility.

The original Unicode standard (1.0) defined a 16-bit entity as the basic unit to represent a character. This became the basis of the UCS-2 encoding of Unicode, which specifies 16 bits per character, regardless of which language it may represent.

However, the UCS-2 standard considered 8 consecutive bits of zero value to be valid data, which has a different meaning to programs written in C—it means the *end of string*. Since most Web and communications software was written in C at the time the Unicode standard was introduced, an alternative encoding of Unicode called UTF-8 became popular. It encodes exactly the same set of characters, but avoids the *null byte* problem. To do this, it represents data in variable amounts—1, 2, or 3 bytes in length, depending on the character.

Today the Unicode standard has advanced further, and has defined an extension mechanism to encode more than 16 bits worth of information. This revised standard is now referred to as UTF-16. The UTF-8 standard has remained popular among Web users, and has added a fourth byte in size to address the Unicode extension mechanism. This means that today there are two forms of Unicode in active use, UTF-16 and UTF-8, and Siebel applications use both of them.

For more information about databases and character sets supported by Siebel applications, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

UCS-2

UCS-2 stands for Universal Character Set - 2 Bytes. In this standard, all characters are represented by two bytes (16 bits), no matter the origin.

UTF-8

UTF-8 stands for Unicode Transformation Format, 8-bit Encoding. UTF-8 is an encoding of Unicode which is more efficient for storage of English (ASCII), whereas other language data is expanded and can be represented by up to four bytes.

For example, English (ASCII) characters use one byte per character, accented European characters use two bytes, and Asian languages use three bytes per character.

UTF-16

UTF-16 replaces the original UCS-2.

UTF-16 can access 63,000 characters as single Unicode 16-bit units and an additional one million characters through a mechanism known as surrogate pairs. For surrogate pairs, two ranges of Unicode code values are reserved for the high (first) and low (second) values of these pairs. High values are from 0xD800 to 0xDBFF, and low values are from 0xDC00 to 0xDFFF. The number of characters requiring surrogate pairs should be fairly limited, because the most common characters have already been included in the first 64,000 values.

About Character Set Encodings and Siebel Applications

Siebel character set encodings are used in multiple places in Siebel applications.

- **Enterprise DB Server Code Page system preference.** This system preference is set during Siebel Enterprise Server installation to reflect the character set that the administrator believes has been set up in the database. *This value must not be modified*, because it is used at installation time to select the correct database schema to be used. (Siebel Systems provides customized schemas to match each database and character set.)

  For more information, see the *Siebel Installation Guide* for the operating system you are using.

- **SIEBEL_CODEPAGE (UNIX environment variable).** This environment variable is created and set by Siebel Systems to indicate the code page that Siebel applications should assume if Siebel configuration files (CFG, CSS, SWT files, and so on) have *not* been saved as Unicode UTF-8—as they would normally be saved. This variable generally does not need to be set explicitly. If you must set it, the value can be a subset of character set encodings, except UTF-8 and UTF-16.

  For more information, see the *Siebel Installation Guide for UNIX: Servers, Mobile Web Clients, Tools*.

- **Character conversion argument.** This argument is available in the following business services:

  - **Transcode Service business service.** Accepts all supported character set encoding names. This business service is normally used for data validation and to prevent data that cannot be converted to the appropriate code page from entering or leaving the Siebel application.

    NOTE: Whenever possible, customers should use the EAI business service or the XML Converter business service to convert data.

  - **EAI business service (MQ Series, DLL, File, HTTP, MSMQ).** Accepts a subset of character set encodings.

  - **XML Converter business service.** Accepts a large number of character set encodings.

  When business services are invoked from a workflow, the valid set of encodings is controlled by a picklist. If the business services are invoked through scripting or a similar mechanism, the character set name is supplied textually.

Special Character Set Encodings and Business Services

In addition to the character set encodings (code pages) listed in *System Requirements and Supported Platforms* on Siebel SupportWeb, some Siebel business services take special character set encodings as inputs.

Updating Currency Symbols

In some situations, you may need to update your currency symbols. For example, if you are operating in a Unicode environment, but your currency seed data was originally installed in a non-Unicode environment, you must update your currencies to include any currency symbols you require that were not part of your prior non-Unicode environment.

For information about activating and defining currencies, see the *Applications Administration Guide*.

About the Database Collation Sequence

The collation sequence, also called sort order, is the ordering relationship, or sequence, between data records. Each database has a collation sequence so that records returned by queries can be returned in a certain order, such as an alphabetic order for text strings. The collation sequence determines the order in which records are displayed in the Siebel client, most noticeably in list views.

A collation sequence is defined when you set up the Siebel Database. All sorting is done in the Siebel Database by the database server. Sorting is not set or performed within the Siebel application and does not depend on the operating system.

NOTE: For more information about creating and configuring the Siebel Database, see the *Siebel Installation Guide* for the operating system you are using. For more information about collation sequences for upgrade environments, see the *Upgrade Guide*.

For the collation sequences supported for each supported RDMBS platform for the Siebel Database, see *System Requirements and Supported Platforms* on Siebel SupportWeb. Also consult your RDBMS vendor documentation.

The collation sequence in effect for a database is determined by one of the following implementation methods:

- Indexes created in the Siebel Database provide a default collation sequence. In Oracle databases, indexes always use binary collation sequence.
- Post-query sorting may also be supported for an RDBMS platform. However, this method of sorting yields slower performance and requires all records to have been retrieved first. For this reason, it is impractical for Siebel applications, which always perform open-ended queries.

For the development environment, only binary collation sequence is supported. For a production environment, you can specify the collation sequence most suitable for your deployment.

NOTE: Changing the collation sequence after the Siebel Database has been installed requires rebuilding your indexes. On a fully loaded production database, this task is time-consuming and database resource-intensive. It is advisable to consult with Siebel Expert Services when planning a project of this complexity.

Which collation sequence is best for your deployment depends on factors such as RDBMS support, performance requirements, database availability requirements, the code page in use, the needs of your users, and the nature of the data that is to be retrieved by different groups of users.

Binary collation sequence offers the best performance and does not require you to rebuild your indexes for the production environment. This collation sequence works well for users working with English-language data, because the ASCII character set is based on the English alphabet and corresponds to the binary collation sequence. However, sorting may be unsuitable for users and data in languages other than English.

For multilingual deployments using Unicode, a linguistic collation sequence based on the Unicode Collation Algorithm (UCA), which goes by different names for different RDBMS vendors, may be a suitable collation sequence. UCA, also known as ISO 14651, provides reasonably good results with mixed-language data.

Other linguistic, or dictionary, collation sequences may offer optimal sorting results for particular languages or groups of languages. Such collation sequences may be suitable for certain deployments, such as those requiring compatibility with the CP932 (Japanese Shift-JIS) sort order.

Linguistic collation sequences that are not based on UCA may associate multiple characters (such as accented and unaccented versions of a particular letter) so they will be treated the same for sorting purposes, but will also be treated the same in unique indexes. If you are changing to a case- or accent-insensitive collation sequence, you will need to first clean out any data that is unique only due to a case or accent difference.

Database Collation for the Mobile Web Client

For the Siebel Mobile Web Client, which uses a local SQL Anywhere database, the default collation sequence for local database indexes is determined by the setting for the local database template from which each individual mobile user's local database is initialized.

Binary, UCA, and other linguistic collation sequences are available for the local database. Binary collation sequence provides the best performance, but other collation sequences may perform acceptably.

NOTE: Synchronization conflicts could occur when collation sequences with different case or accent sensitivity are in effect on the local database and the enterprise database. For this reason, you should choose a collation sequence on the local database that is at least as case or accent sensitive as the collation sequence on the enterprise database. Avoid choosing a case-insensitive collation sequence on the local database if your server database is case sensitive.

It is possible to specify a post-query collation sequence for an individual mobile user's local database, by specifying a value for the SORTCOLLATION parameter in the application configuration file. Performance issues may arise from specifying a post-query collation sequence in this manner.

For more information about the Mobile Web Client and the SORTCOLLATION parameter, see the *Siebel System Administration Guide*, the *Siebel Remote and Replication Manager Administration Guide*, and other applicable documentation on *Siebel Bookshelf*.

NOTE: A local database used for development with Siebel Tools must use the binary collation sequence. Using Siebel Tools against a non-binary collation sequence is not supported.

3 Planning Global Deployments

This chapter provides high-level guidelines in how to successfully plan a global deployment. It includes the following topics:

Understanding Your Company's Global Business Requirements

Imagine that your company tells you that it wants to market the software it develops to four other countries and wants the applications to run in the languages of those countries. How do you start? What do you need to consider to make sure that the new product development effort is successful? What do you need to think about when customizing Siebel Business Applications for this purpose?

Although you must take your company's unique business requirements into consideration in your planning, this guide offers tips and guidelines for undertaking a global product rollout and maintenance in general, as well as configuring the Siebel application in particular.

By addressing the following questions concerning your organization's global business needs, you will have gone a long way towards planning for your global deployment of Siebel Business Applications:

- Will your company have one central business location from which all business transactions originate, or regional decentralized (distributed) locations for transactions?
  - If decentralized, does your company need to keep the transactions synchronized, for example, banking transactions?
  - Will remote users synchronize their transactions to a central corporate database or a regional database?
- Are CTI, Siebel Report Servers, email servers, Siebel Analytics, and other servers centralized or regional?
- Which languages does your company headquarters require?

- Which language will be the base, or primary, language with which you begin your development and customization process? The first language installed is the primary language.
- Have you previously customized any language files from a previous version of Siebel Business Applications, or configured a new language not offered yet by Siebel Systems? Languages previously provided by Siebel Systems can be upgraded. If you previously localized into a language that Siebel Systems now provides directly, you must either merge your previous use of this language with the Siebel language, or keep them separate.
- Which locales are the languages intended for (for example, French-speaking Canada or France)?
  - Which locale settings will be needed as a result?

    The answer to this will affect the way currency, numbers, dates, and times are formatted in the software.
  - Because additional Siebel Application Object Managers will be needed for each locale/language, determine in advance the implications for memory and performance of the products you use.
- What character set will you use for your database—a Unicode or a non-Unicode character set?

  This decision has far-reaching implications for the ease with which your organization can deploy globally. If you do not implement a Unicode database, then you may not be able to support all languages which your business uses. In that case, you won't be able to roll up data from those countries into your Siebel applications, or into a data warehouse.
- Are there legacy interfaces that you need to consider in your planning and do these have implications for your back-office applications? In which code page is the data of your back-office applications expected? Is there a need to convert between code pages, such as from Unicode to non-Unicode?
- Who will localize your customizations?
- Do you have particular legal requirements you must meet within your global network (for example, as regards European Union Data Protection Directives, Basel II, or others)?

Expected Results of Global Deployment Planning

The answers to the questions in “Understanding Your Company’s Global Business Requirements” on page 25 should help determine your:

- Globalization strategy
- Globalization project timeline
- Network diagram
- Capacity planning
- Call handling strategy for CTI and call center features
- Localization scope, for example, for the List of Views and Applets by language
- Localization budget

- List of modifications you need for preconfigured functions required by a specific locale
- List of tables and data to be exchanged with other applications, as determined from your analysis of any legacy interfaces

By considering the questions listed in the foregoing sections before you undertake a global deployment, you stand to decrease the total cost of ownership (TCO) your company pays for its global outreach, in the form of:

- Driving down the cost of localization development
- Decreasing global deployment costs in general
- Shortening the time to market for global deployments

About Ideal Global Deployments

Figure 2 on page 28 provides an example of what an ideal Siebel global deployment might look like. Keep in mind that few companies necessarily meet an ideal. However, with proper planning, the ideal is a goal that can be achieved.

In this example, six languages (Language Packs) have been installed on the Siebel Server. Different Application Object Managers running on this Siebel Server can display the application user interface in these languages.

This Siebel Database uses Unicode and therefore supports all characters required for the six languages installed on the Siebel Server. The other two languages shown, Chinese (CHS) and French (FRA), must be supported for customer data in this example deployment, even though these two languages are not used for the Siebel application user interface.

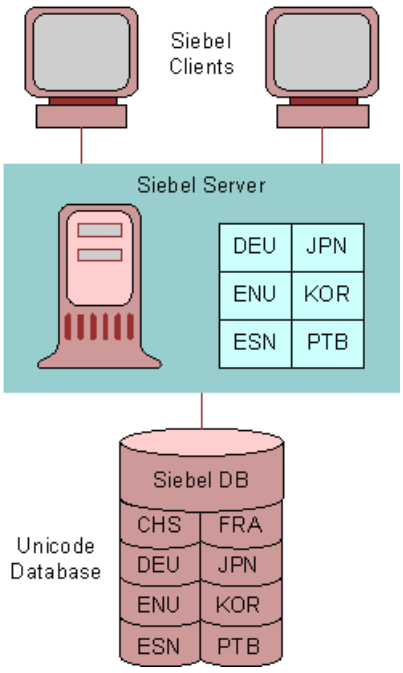

Figure 2. Example of an Ideal Global Deployment

Figure 2 on page 28 illustrates an ideal centralized global deployment running with a Unicode database for Siebel applications (for version 7.5 or greater). Unicode is the character set of the database in this ideal configuration.

For a production environment, you can specify the collation sequence most suitable for your deployment. For the development environment, only binary collation sequence is supported. For more information, see "About the Database Collation Sequence" on page 23.

Where necessary, character set conversions take place to and from all Siebel (Unicode) and non-Siebel data sources.

Siebel clients run in this deployment on localized operating systems with Unicode fonts installed for selected users. The user interface appears in the language of the user's choice. Even languages such as Japanese and Arabic are encoded properly.

For more information about installing languages, see the *Siebel Installation Guide* for the operating system you are using.

For your global deployment, Universal Time Coordinated (UTC) and multilingual lists of values (MLOVs) should be enabled.

For more information about deploying with UTC, see Chapter 5, "Deploying with Global Time Zone."

For more information about configuring LOVs and MLOVs, see "Localizing Lists of Values and Multilingual Lists of Values" on page 84.

The ideal global configuration allows deployment of Siebel Business Applications to satisfy the requirements for either a centralized or a decentralized (distributed) global enterprise.

A centralized global deployment requires only a single Siebel installation, and consolidates all customer-related information in one data store, creating a single, global view of customers transactions. A centralized deployment is easier and cheaper to maintain and may be satisfactory, depending on your business needs.

A decentralized (distributed) global deployment may have Siebel instances in the Americas, Europe, and Asia, as shown in Figure 3 on page 29. Data is replicated between these sites using Siebel Replication Manager. For more information, see the *Deployment Planning Guide*, the *Siebel Remote and Replication Manager Administration Guide*, and other applicable documentation.

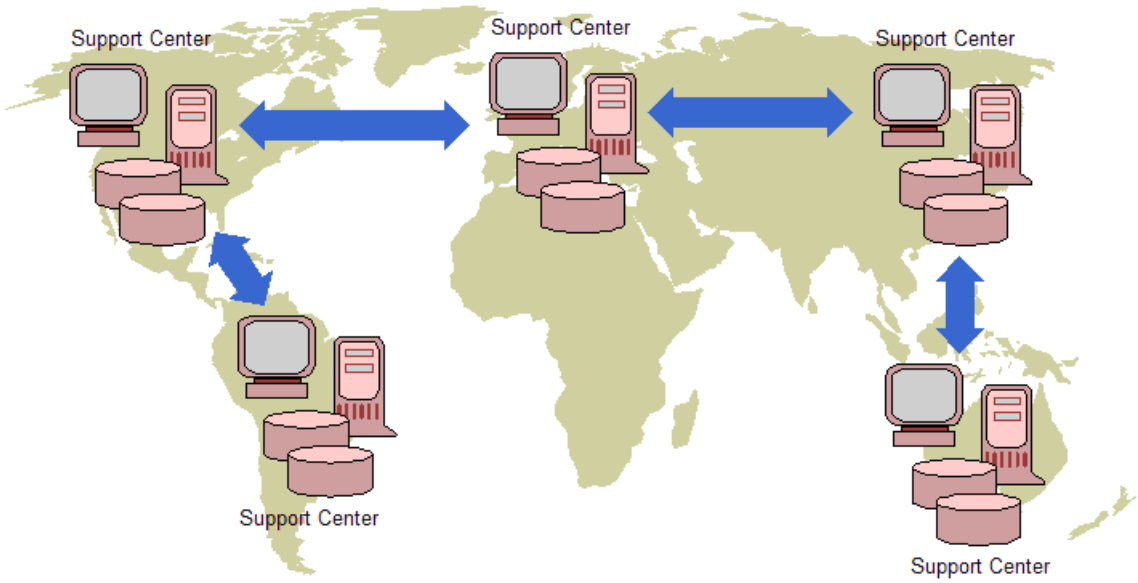

Figure 3. Example of a Decentralized Siebel Global Enterprise

About the Siebel Unicode Architecture

The Siebel software architecture uses Unicode, including for internal processing. There are many benefits to using Unicode, such as the ability to show multilingual data in the same session. Figure 4 on page 30 provides a graphical representation of the architecture, while Table 4 on page 30 describes the flow of data through the Siebel Unicode architecture.

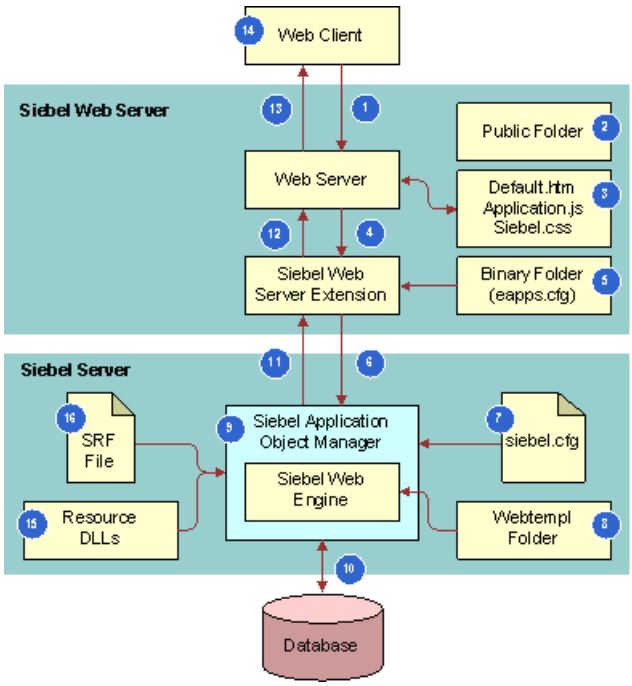

Figure 4. Siebel Unicode Architecture

Table 4 on page 30 describes the flow of data through the Siebel Unicode architecture. For more information about supported databases, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

Table 4. Siebel Unicode Architecture Data Flow

Step Number	Text Encoding
1	User input is URL-encoded by JavaScript or browser.
2	JavaScript code and style sheets are in ASCII. **NOTE:** Style sheet files can be saved as UTF-8 in order to be able to specify font names that contain non-ASCII characters.
3	HTML pages are in ASCII, or UTF-8 if hard-coded non-ASCII content is required.
4	Web server passes URL-encoded input to Siebel Web Server Extension (SWSE).
5	SWSE configuration file (eapps.cfg) is in ASCII.
6	SWSE sends input (still in URL encoding) to the Siebel Server. The Siebel Server decodes input immediately to the Siebel internal encoding (UTF-16).
7	Application configuration file (CFG) is UTF-8. (Some parameter values may specify non-ASCII text for elements such as font names or splash screen text.)

Table 4. Siebel Unicode Architecture Data Flow

Step Number	Text Encoding
8	Web templates are in ASCII and are language-independent.
9	Most C++ code uses the Siebel internal encoding (UTF-16).
10	Application Object Manager uses UTF-16 to communicate with the database. The internal encoding of the database can be UTF-8, UTF-16, or a code page.
11	Siebel Web Engine (SWE) sends the HTML page (in UTF-16) to SWSE.
12	SWSE transcodes, or converts, the HTML page to UTF-8 for the World Wide Web.
13	Web server passes the HTML page (in UTF-8) to the browser.
14	Browser code (JavaScript or Java) reads the HTML page in UTF-16.
15	Resource dynamic-link libraries (DLLs) store content in UTF-16.
16	Siebel repository file (SRF) stores content in UTF-16.

About Planning Your Global Deployment

As you begin to evaluate your global deployment needs, start with the following steps:

1. Determine what your base application language will be. In general, this book assumes a base application of American English (ENU).
2. Consider available industry-specific products and the operating system you use.
3. Determine what needs to be localized. For example:
   - Menus
   - Picklists
   - Lists of values
   - View names
   - Strings in applets
   - Reports
   - Correspondence templates
   - Communications templates (for example, for Siebel Email Response, Send Email, or Send Fax)
   - Forecasts
   - Personalization rules
   - Workflow policies
   - Assignment rules
   - Currency

- Master data (for example, product and catalog data)
- Online Help

  Your Siebel application includes localized online help. Localized online help files are located in the language-specific directories on either the server or the Mobile or Developer Web Client. For information about how online help is implemented or how to customize online help, see *Configuring Siebel Business Applications*.

- iHelp

  For more information about creating and administering iHelp, see the *Applications Administration Guide*.

4. Perform a gap analysis to determine functionality that needs to be improved or turned off.

   You may find that you need to perform additional steps to enable multilingual support. For example, you might want to enable multilingual lists of values to support multilingual picklists. In this case, evaluate your Siebel Business Applications' performance for columns used in search specifications.

5. Configure your application based on your company's business requirements.
6. Deploy the application to a global user base by doing the following:

   - Install the necessary Siebel Server, Siebel Gateway Name Server, Siebel Web Server Extension, and Language Packs for your production system. The Siebel Language Packs will install the language-specific run-time environment: repository (SRF) files, DLLs, configuration (CFG) files, reports, error messages, help files, and so on.
   - Install the necessary language-specific seed data into the Siebel Database.

     For more information, see the *Siebel Installation Guide* for the operating system you are using.

   - Deploy the Siebel Mobile Web Client in the local language by installing the necessary Language Packs.

     For more information, see the *Siebel Installation Guide* for the operating system you are using, the *Siebel Anywhere Administration Guide*, and the *Siebel Remote and Replication Manager Administration Guide*.

   NOTE: Many tasks for multilingual deployments are more complex and time-consuming than they are for a single-language deployment. Such tasks include installing applications, localizing custom configurations, administering data in multiple languages (MLOV and master data), and training users. Depending on your requirements, additional hardware may be needed to support deploying multilingual applications globally.

About Configuring Global Deployments

By answering the following questions, you will be underway to planning for configuration of your global network applications. For detailed information about configuration tasks, see Chapter 4, "Configuring Global Deployments."

- Which address applet layouts do you want to modify for your target countries?

- Which Name layouts do you want to modify for your target country and languages?
- Do you want to add *preferred language* to contact records? This is the language in which one of your customers prefers to receive communications such as email messages, brochures, and so on.
- Do you want to separate prospect lists by country and language?
- Do you want to add language/country code to attachment names? Doing so may help you identify attachments intended only for an audience using a particular language or page size, for example.
- Do you want to extend product masters, training course descriptions, and other master data to have local language versions? Doing so may help you provide such information to the right audience.
- Do you want to mark selected data records with a country or language tag? Doing so may help you roll up sales records originating in a particular country or region, for example.
- Have you converted LOVs to MLOVs, where appropriate?
- Have you updated LOVs or MLOVs with international content—for example, lists of countries, offices, and other information? Have you previously added organizations by country and/or region? Organizations in the Siebel application may be defined as countries. See also "Localizing Lists of Values and Multilingual Lists of Values" on page 84.
- Have you created localized response templates for email?
- Have you created multilingual PDQs? This term refers to PDQs that are constrained by language or by country, in order for the query names to display using the appropriate language.
- Have you modified custom search criteria to match non-English records?
- Have you created localized seed and demo data?
- Have you modified workflows to perform LOV value lookup?

By addressing the foregoing questions, you achieve the following outcomes:

- Address and name applets formatted to your target countries
- Successfully configured LOVs or MLOVs, and PDQs
- Successfully configured email servers and other third-party servers
- Successfully localized seed and demo data
- Workflows modified for language and locale
- Successfully localized global data, iHelp, SmartScripts, online help, and so on

Options for Installing Multiple Language Packs on Siebel Servers

Consider the following deployment options when you plan to install multiple languages on your Siebel Servers.

You can install Language Packs at any time to meet business needs. However, there are advantages to installing as many as you can during the initial configuration. For details, see the *Siebel Installation Guide* for the operating system you are using.

1. "Deploying Language Packs Only Once" on page 34. For users deploying only those languages shipped with the current release of Siebel Business Applications who do not plan to deploy other languages at a later date.
2. "Deploying Language Packs Now and at a Later Date" on page 34. For users deploying only those languages shipped in the current release of Siebel Business Applications who plan to deploy other languages at a later date.
3. "Deploying Language Packs Shipped After Initial Rollout" on page 35. For users deploying languages shipped in a Siebel Business Applications 7.8.x release subsequent to the release installed by the customer.

Deploying Language Packs Only Once

You are deploying only those languages shipped with the current release of Siebel Business Applications and you are installing all the languages your enterprise will require during initial installation and configuration of your Siebel Servers.

To deploy multiple Language Packs shipped with the current release

1. Install all Siebel Servers you require with all the Siebel Language Packs intended to run on each physical machine.
2. Configure each Siebel Server, as prompted.
3. On each Siebel Server, disable any language-specific Application Object Managers (AOMs) that you do not want to run automatically on that server.
4. Install language-specific seed data in the database.

For more information about the above steps, see the *Siebel Installation Guide* for the operating system you are using.

Deploying Language Packs Now and at a Later Date

The following procedures are intended for users who are deploying only those languages shipped in the current release of Siebel Business Applications, but who plan to deploy some languages at a later date. There are two possible ways to achieve this. Review each procedure and decide which option works best for your situation.

To deploy multiple Language Packs now and at a later date (Option 1)

1. Install all intended languages at once.
2. Complete the task "Deploying Language Packs Only Once" on page 34.
3. Also disable any language-specific AOMs that you intend to use in a future deployment.
4. Re-enable the language-specific AOMs that you disabled in Step 3 on page 35 when you are ready to deploy them.

For more information about the above steps, see the *Siebel Installation Guide* for the operating system you are using.

To deploy multiple languages now and at a later date (Option 2)

1. Install initial rollout languages.
2. Complete the task "Deploying Language Packs Only Once" on page 34.
3. Complete the task "Deploying Language Packs Shipped After Initial Rollout" on page 35 when you are ready to deploy additional languages.

Deploying Language Packs Shipped After Initial Rollout

The following procedure is intended for users deploying languages shipped in a Siebel Business Applications release 7.8.x subsequent to the release installed by the customer. You can either:

- Add new languages to your existing physical resource allocation. For example, you might install a new Siebel Server with a language you did not previously install elsewhere.
- Add new physical resources as part of expanding language support. For example, you might install a new language on an existing Siebel Server.

NOTE: For more information about these scenarios, see the *Maintenance Release Guide* for version 7.8.x, which is available on Siebel SupportWeb.

To add new Language Packs to your existing physical resource allocation

1. Complete the task "Deploying Language Packs Only Once" on page 34.
2. Run the `srvrcfg` command to import the Application Object Manager (AOM) definitions file from the newly installed Siebel Language Packs on the servers where they will be deployed. This step creates language-specific AOM components for the languages you added.

   CAUTION: Before you run the `srvrcfg` command, you must stop the Siebel Server. However, the Siebel Gateway Name Server must be running.

   - Windows: Enter the following command at a DOS prompt from within the *SIEBSRVR_ROOT*\bin directory of your Siebel Servers:

     ```
     srvrcfg /g SiebelGatewayNameServer /e SiebelEnterprise /a components /i
     SIEBSRVR_ROOT\bin\LANG\omdefs_file /c ALL
     ```

- UNIX: Enter the following Korn or Bourne shell command from within the *SIEBSRVR_ROOT*/bin directory of your Siebel Servers:

  `./srvrcfg -g` *`SiebelGatewayNameServer`* `-e` *`SiebelEnterprise`* `-a components -i` *`SIEBSRVR_ROOT`*`/bin/`*`LANG`*`/`*`omdefs_file`* `-c ALL`

  where:

  SiebelGatewayNameServer = The alias of your Siebel Gateway Name Server

  SiebelEnterprise = The alias of your Siebel Enterprise Server

  SIEBSRVR_ROOT = Your Siebel Business Applications installation directory

  LANG = The three-letter code for the language you are implementing

  omdefs_file = The file from which you are importing Object Manager definitions:

  - For Siebel Business Applications, use omdefs.dat
  - For Siebel Industry Applications, use omdefs_sia.dat

3. Disable any language-specific AOMs that you do not intend to use on specific servers.

For more information, see the *Siebel Installation Guide* for the operating system you are using.

To add new physical resources as part of expanding language support

1. Complete the task "Deploying Language Packs Only Once" on page 34 while installing new languages on new physical resources.
2. Create new language-specific AOMs as part of configuring each new server.
3. Disable any language-specific AOMs that you do not intend to use on specific servers.

For more information, see the *Siebel Installation Guide* for the operating system you are using.

Planning Upgrades for Global Deployments

This section describes considerations for planning a successful upgrade of Siebel Business Applications within a global deployment.

For detailed information about the upgrade process, see the *Upgrade Guide*.

Planning for the Upgrade

Addressing the questions in the following list will help you prepare for a successful upgrade of Siebel Business Applications within a global deployment.

- What database version do you use?

  Where applicable, you must upgrade your RDBMS to a supported version for Siebel 7.8. If you are moving to Unicode, the new database version may also make this possible.

- What version of Siebel Business Applications will you be upgrading from? Current versions of Siebel Business Applications support Unicode.
- Do you know what is involved in migrating your data to Unicode?
- Did you previously deploy a customized language not certified by Siebel Systems? Such languages are also referred to as *unshipped languages*. See also "Localizing an Unshipped Language" on page 85.
- Do you know how to upgrade date-time fields to Universal Time Coordinated (UTC)? See also Chapter 5, "Deploying with Global Time Zone."
- Are the Siebel Language Packs you require released and available? What Siebel release levels (including patches) are required for implementing these languages?
- Have you upgraded client machines (including keyboards, fonts, locales installed, and localized operating system) to support the new installed languages?
- Have you evaluated the need to upgrade all third-party products to versions that support Unicode, the countries/locales you will be localizing for, and the languages you require?

Results to Expect After Upgrading

After you have addressed the issues above and performed the necessary upgrade tasks, you should have gone through the following conversion processes:

1. Updating databases and Siebel versions to current versions.
2. Converting data to Unicode or to another code page that is supported with the newer versions of Siebel applications. (Conversion to Unicode requires the assistance of Siebel Expert Services.)
3. Converting selected LOVs to MLOVs.
4. Converting selected date-time data to UTC.

4 Configuring Global Deployments

This chapter discusses how to configure your Siebel Business Applications for a global deployment. It includes the following topics:

About Parameters for Language and Locale

This section describes the parameters that can be set on the Application Object Manager in order to specify language, resource language, and locale.

On the Siebel Mobile Web Client, equivalent parameters may be set in the application configuration file (such as uagent.cfg for Siebel Call Center).

NOTE: If you are localizing an unshipped language, see also "Localizing an Unshipped Language" on page 85. In particular, for scenarios for setting the language parameters, see "Creating Language and Locale Records" on page 86 and "Creating Application Object Manager Components" on page 90.

Language Code

The Language Code parameter (alias Lang) determines the language used for multilingual lists of values (MLOVs) and other application seed data. This parameter also determines which language-specific directory from the product installation will be used for accessing the SRF file. The language of application user interface labels is determined by the language for the SRF file. This language was determined by the Siebel Tools language mode in effect when the SRF file was compiled.

For each language you install, Application Object Manager components are created which are already configured with Language Code set to this language. In general, you should not need to change the value of the Language Code parameter for these components. Special requirements apply when you localize an unshipped language. For details, see "Localizing an Unshipped Language" on page 85.

The Application Repository File parameter (alias CFGRepositoryFile), set for the Application Object Manager, specifies the name of the SRF file. Where appropriate, this parameter can optionally specify the absolute path to the SRF file, in order to use the SRF file in a different language directory—such as to display the user interface in a different language than that specified by Language Code.

Unless the OM - Resource Language Code parameter is also set (to a different language), the Language Code parameter also determines which language-specific directories will be used during runtime operation for resource libraries such as DLLs. These resource libraries determine the language in which system and error messages display and the language in which server log file messages are written.

Each language has its own three-letter code identifier—for example, ENU identifies U.S. English. Languages supported by Siebel Systems are identified in *System Requirements and Supported Platforms* on Siebel SupportWeb. See also Table 8 on page 55.

NOTE: On the Siebel Mobile Web Client, the equivalent parameter is the Language parameter in the [Siebel] section of the configuration file. For each language you install, language-specific directories are created containing the configuration files, in which Language is already set. In general, you should not need to change the value of the Language parameter. Special requirements apply when you localize an unshipped language. For details, see "Localizing an Unshipped Language" on page 85.

The setting of the Preferred Language field in the Contacts screen overrides the value of the Language Code parameter for determining the language for multilingual lists of values (MLOVs). This setting is stored in S_CONTACT.PREF_LANG_ID in the Siebel Database. This field does not affect Mobile or Developer Web Client users.

See also "Localizing Lists of Values and Multilingual Lists of Values" on page 84.

See also the description for OM - Resource Language Code and see "About the Active Language" on page 42.

For more information about the effect of the language mode in Siebel Tools and about how to set it, see *Using Siebel Tools*.

OM - Resource Language Code

The OM - Resource Language Code parameter (alias ResourceLanguage) can optionally be set to a different language than the Language Code parameter in order to specify the resource language.

The resource language determines which language-specific directories from the product installation will be used during runtime operation for resource libraries such as DLLs. These resource libraries determine the language in which system and error messages display and the language in which server log file messages are written.

DLLs are provided for all languages shipped by Siebel Systems. These DLLs cannot be localized into any other unshipped languages.

For some multilingual deployments, it may be useful, or necessary, to set the resource language to a different value than the Language Code parameter. For example, you may want all global users to view MLOV and other seed data in the language determined by Language Code. However, you may choose to set the resource language to a single language familiar to your administrators.

Where Language Code is set to an unshipped language, you must use the DLLs for a language provided by Siebel Systems. In this case, you can set OM - Resource Language Code to specify which language's DLLs to use. For example, where Language Code is set to the unshipped language PLK (Polish), you might set OM - Resource Language to DEU (German). For details, see "Localizing an Unshipped Language" on page 85.

NOTE: On the Siebel Mobile Web Client, the equivalent parameter is the ResourceLanguage parameter in the [Siebel] section of the configuration file.

See also the description for Language Code and see "About the Active Language" on page 42.

Locale Code

The Locale Code parameter (alias LocaleCode) specifies the locale associated with this Application Object Manager (AOM) component. This setting is used by the Siebel Web Clients for this AOM.

The Locale Code setting is a three-letter locale code. A locale is a set of rules guiding how common data is displayed to the user or is received from the user. Siebel Business Applications support formatting of data, such as dates, time, numbers, and currency, based on locale settings.

Locales are administered using the Locale Administration view.

For each language-specific AOM component on a Siebel Server, the Locale Code is set to a locale that may be appropriate for users for that language. If you need to support multiple locales for the same language, then you can use either of the following approaches to deploy your AOMs:

- *On different servers*, configure and run different locale-specific AOMs for this language on different Siebel Servers. For example, an FRA AOM would be running in France with a French locale, while another FRA AOM would be running in Canada with a French Canadian locale.
- *On the same server*, create, configure and run different AOMs that are specific to both this language and a particular locale. For example, one FRA AOM running in France would have a French locale, while another FRA AOM on the same Siebel Server in France would have a French Canadian locale.

If the Locale Code parameter is not set, then the AOM and the Siebel Web Clients use the locale defined in the operating system on the Siebel Server machine. On Windows, this setting is defined in Regional and Language Options in the Control Panel.

NOTE: On the Siebel Mobile or Developer Web Client, the locale is always determined by the settings defined in Regional and Language Options in the Control Panel.

Additional Information About Setting Up and Administering Locales

During installation, you need to set locales for your Siebel Web Clients, Siebel Servers, and database. After installation, you can modify the Siebel Server locale or add additional locales. Table 5 on page 42 shows where you can find detailed information on performing these procedures.

Table 5. Additional Information on Setting Up Locales

For information on ...	See ...
Administering Siebel Server locales	*Applications Administration Guide*.
Setting Siebel Server locales	*Siebel Installation Guide* for the operating system you are using.
Setting Siebel Mobile Web Client and Developer Web Client locales	On a Windows PC, locales are administered through settings found on the Control Panel.

About the Active Language

The active language for Siebel Business Applications is the language used by the application, as determined by the following:

- **Siebel Server/server components.** The Language Code parameter setting for the Siebel Server determines the language in which server log file messages are written. Override values for this parameter are set for many server components, including Application Object Managers.

  For more information, see the *Siebel Installation Guide* for the operating system you are using and the *Siebel System Administration Guide*.

  For deployment scenarios for installing multiple languages on Siebel Servers, see "Options for Installing Multiple Language Packs on Siebel Servers" on page 34.

- **Siebel Web Clients.** The active language for a user session in the Siebel Web Client is determined by the Language Code parameter setting for the Application Object Manager component to which this client is connected.

  For more information, see the *Siebel Installation Guide* for the operating system you are using and the *Siebel System Administration Guide*.

  See also "About Parameters for Language and Locale" on page 39.

- **Siebel Mobile Web Clients**. The active language for a user session in the Siebel Mobile Web Client is determined by the Language parameter setting in the application configuration file. There are different configuration files for each language. They are located in a language-specific directory and have the Language parameter set appropriately. For example:

  - C:\Program Files\Siebel\7.8\Web Client\bin\deu\uagent.cfg (configuration file with Language parameter set to DEU)
  - C:\Program Files\Siebel\7.8\Web Client\bin\enu\uagent.cfg (configuration file with Language parameter set to ENU)

Do not modify the Language parameter to change the active language. Instead, the user should specify the appropriate application configuration file for the language in which to run, by using the /c switch in the command line of the siebel.exe application shortcut.

For example, to run Siebel Call Center in German, the properties of the Siebel application shortcut created during installation with the DEU language include something like this:

```
/c C:PROGRA~1\SIEBEL\7.8\WEBCLI~1\bin\DEU\uagent.cfg
```

For more information, see the *Siebel Installation Guide* for the operating system you are using and the *Siebel Remote and Replication Manager Administration Guide*.

NOTE: Siebel applications use Microsoft codes for Siebel language specifications. For information about Microsoft codes, see your Microsoft documentation.

- **Siebel Tools**. In Siebel Tools, you set the language mode to work with object definitions for a particular language. The language of an SRF file is determined by the language mode in effect when the SRF file was compiled.

  For more information about the effect of the language mode in Siebel Tools and about how to set it, see *Using Siebel Tools*.

About Configuring Language Support for Browser Platforms

A browser platform includes the following:

- Web browser
- Java runtime environment used by the Web browser (plug-in)
- Operating system where the Web browser runs

For a list of supported Web browsers, operating systems, and other client environment requirements, see *System Requirements and Supported Platforms* on Siebel SupportWeb. See also the browser configuration chapter in the *Siebel System Administration Guide*.

Current Web browser platform default configurations are not installed with all fonts and other resources necessary to display all Unicode characters outside the default non-Unicode code page. Some manual setup may be required in order to view all characters correctly in Web pages, including Siebel Web Client screens. More recent versions of operating systems, browsers, and Java runtime environments require less setup than earlier versions.

If characters from some languages appear as hollow or solid boxes, it may be due to lack of language setup on the browser platform. Box characters always indicate a problem at the user interface level. Question marks appear when Unicode characters are mishandled on the server or within third-party applications. They can also be due to the character set having been incorrectly configured in the database.

While the vendor's documentation and support should be definitive on these issues, the following hints can provide a starting point for resolving any issues.

There are several contexts where Web browsers display text, including:

- HTML body text (most text in Siebel client screens)
- HTML field input (all text input in Siebel client screens)
- ToolTips
- Message boxes (alerts)
- Java applets (used in messages, toolbars, popup calculators, and calendars)

This section contains the following topics:

About Unicode Fonts

This book sometimes mentions *Unicode fonts*. This term refers to a font that contains glyphs for most, if not all possible characters in Unicode. Such fonts are not shipped as standard parts of any operating system. Customers must license them or otherwise obtain them for installation on their client machines.

A commonly used font of this type is Arial Unicode MS, which comes with every installation of Microsoft Office. However, it must be installed through an additional procedure (see Microsoft Office documentation), and it only includes glyphs for about 40,000 characters (it is compatible with Unicode 2.0).

See also "Configuring Cascading Style Sheets to Specify Different Fonts" on page 54.

Configuring Language Support for Windows 2000

If your end users run Windows 2000, complete the following procedures to configure language support:

- Installing language support and enabling text display
- Adding input locales
- Configuring ToolTips and message boxes
- Configuring Java fonts

Installing Language Support for Windows 2000

Complete the following procedure to install language support for Windows 2000. This procedure also allows the correct display of body text.

To install language support for Windows 2000

1. Choose Start > Settings > Control Panel.
2. Double-click Regional Options and click the General tab.
3. Select all of the Language settings for the system boxes and click OK.
4. At the prompt, navigate to the Windows install CD or network location to copy Windows install files from.

   Allow Windows to restart if requested.

Adding Input Locales for Windows 2000

Complete the following procedure to add input locales for Windows 2000.

To add input locales for Windows 2000

1. Choose Start > Settings > Control Panel.
2. Double-click Regional Options and click the Input Locales tab.
3. In the Installed input locales list, install your desired input locales by selecting them and clicking Add.
4. In the Hot keys for input locales list, set up any desired keyboard shortcuts to switch input locales.

After you install input methods, a tray icon for controlling input locale appears in the lower-right area of the Windows taskbar. To switch input locales for each application, you can left-click the tray icon or use the keyboard shortcuts you set up to switch input locales.

Configuring ToolTips and Message Boxes for Windows 2000

Complete the following procedure to configure ToolTips and message boxes in Windows 2000.

To configure ToolTips and message boxes for Windows 2000

1. Choose Start > Settings > Control Panel.
2. Double-click Display and click the Appearance tab.
3. From the Item drop-down list, select ToolTips.
4. From the Font drop-down list, select a Unicode font (for example, Arial Unicode MS).
5. From the Item drop-down list, select Message Box.
6. From the Font drop-down list, select a Unicode font (for example, Arial Unicode MS) and click OK.

NOTE: These global changes will affect the appearance of all applications and the Windows desktop.

Configuring Java Fonts for Windows 2000

Complete the following procedure to configure Java fonts for Windows 2000.

To configure Java fonts for Windows 2000

1. Install a Unicode font, such as Arial Unicode MS, on Windows 2000.
2. Run regedit and create the following key:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Java VM\Font Alias
   ```

3. Create string values with the name of a Java font as name, and the name of a Windows Unicode font as value. Java has five font names.

   It is also possible to create a REG file with the following text and run it to install the registry entries.

   ```
   REGEDIT4

   [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Java VM\Font Alias]

   "Dialog"="Arial Unicode MS"

   "DialogInput"="Arial Unicode MS"

   "Courier"="Arial Unicode MS"

   "Helvetica"="Arial Unicode MS"

   "TimesRoman"="Arial Unicode MS"
   ```

Configuring Language Support for Windows XP

If your end users run Windows XP, complete the following procedures to configure language support:

- Install language support for Windows XP
- Add input locales for Windows XP

NOTE: By default, ToolTips and message boxes should be properly configured for Windows XP. If they are not, the procedure to configure them is the same as the one for Windows 2000 found in "Configuring Language Support for Windows 2000" on page 44.

Installing Language Support for Windows XP

Complete the following procedure to install language support for Windows XP. This procedure also facilitates the correct display of body text.

To install language support for Windows XP

1. Choose Start > Settings > Control Panel > Regional and Language Options.
2. Click the Languages tab.

3 In the Supplemental language support list, select the following boxes:

- Install files for complex script and right-to-left languages (including Thai)
- Install files for East Asian languages

4 Click OK.

5 At the prompt, navigate to the Windows install CD or network location to copy Windows install files from.

Adding Input Locales for Windows XP

Complete the following procedure to add input locales for Windows XP.

To add input locales for Windows XP

1 Choose Start > Settings > Control Panel > Regional and Language Options.

2 Click the Languages tab, and then click Details.

3 In Installed services, select the desired input services and then click Add.

4 Click Key Settings to set up any desired keyboard shortcuts to switch input locales.

5 Click Language Bar to adjust language bar settings, if necessary.

After you install input services, a dockable toolbar for controlling input languages appears. To switch input locales for each application, you can click the language bar or use the keyboard shortcuts you set up to switch input locales.

Configuring Language Support on Windows ME/98

If your end users run Windows ME/98, complete the following procedures to configure language support.

These operating systems support only the standard interactivity mode for Siebel Web clients, such as is used by customer applications. They do not support high interactivity mode.

NOTE: The Windows ME/98 operating systems do not have cross-code page support, so in this case, Internet Explorer itself can be set up with language support, as described below.

To configure language support on Windows ME/98

1 In Internet Explorer, navigate to Windows Update (http://windowsupdate.microsoft.com).

NOTE: You can also access Windows Update from the Internet Explorer menu bar by choosing Tools > Windows Update.

2 Select the desired language support.

It is not necessary to also install a localized user interface for each language.

NOTE: You may also receive a prompt to install language support from Windows Update when you load a Web page containing characters that are not covered by installed language support.

About Integration Considerations

There are many issues to consider when planning application integration for a global deployment. This topic discusses the following global deployment integration issues:

- "About Character Conversion Errors" on page 48
- "About the Transcode Service Business Service" on page 49

For detailed information about application integration for global deployments, see *Overview: Siebel Enterprise Application Integration* and other documentation for Siebel Enterprise Application Integration (EAI).

For supported character set encodings for supported languages, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

About Character Conversion Errors

When the Siebel application sends data to an external application using a non-Unicode (traditional) character set encoding, the external application may not be able to accept all of the data if the data does not belong to the character set encoding of the application. In this situation, unrepresentable characters will be converted to a substitution character.

The substitution character may be a question mark or an upside-down question mark (indicating character loss in the database), depending on the database. This is referred to as *unrepresentable character conversion*.

The following section describes how to avoid this situation, using techniques for character conversion error handling.

The transport and adapter business services have two character conversion error handling modes:

- Generate an error without sending the message
- Ignore the error, substitute replacement characters wherever possible, and send the message

These error handling modes are specified in an argument to the business service call.

A second case occurs when you are using a Unicode communication stream, such as XML, to communicate between Siebel Business Applications and an external, non-Unicode application. In this situation, the character conversion can be handled by the external application after it receives the message. This could be too late and the external application might not provide the desired error handling functionality.

To avoid this problem, you can use the Validate method of the Transcode Service business service, before sending any data to the external application. This method detects whether character conversion performed by the external application will fail. If it will fail, you can choose not to send the data. For details, see "About the Transcode Service Business Service" on page 49.

About the Transcode Service Business Service

To supplement the character conversion performed by Siebel EAI components, Siebel Business Applications offer the Transcode Service business service. You can use methods of this business service in your workflow processes to validate and convert data from one character set encoding to another.

For information about how business services handle character conversion errors, see "About Character Conversion Errors" on page 48.

The Transcode Service business service supports two methods:

- **Validate.** This method confirms the input property set hierarchy or the *value* of the input property set. You can use this method to check that a character is valid within a particular character set before proceeding, without actually performing the conversion. The Validate method is described in Table 6 on page 49.

Table 6. Arguments for Validate Method

Method Argument	Required	Description
ValidationMode	Yes	If set to *value*, then only the value is validated. Otherwise, the properties are validated for the entire hierarchy.
TargetEncoding	Yes	For the proper language-independent code parameter values, see *System Requirements and Supported Platforms* on Siebel SupportWeb.
SourceEncoding	No	This argument is required when ValidationMode is set to *value* and the input value contains binary data. This implies conversion from binary data in SourceEncoding to binary data in TargetEncoding. For the proper language-independent code parameter values, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

- **Convert.** This method converts the *value* in the input property set to the target encoding in the output. You use this method when data is entering or leaving Siebel applications, and a conversion is required now, so that the next software component in the processing chain can recognize the data. The Convert method is described in Table 7 on page 50.

Table 7. Arguments for Convert Method

Method Argument	Required	Description
ConversionMode	Yes	Values can be StringToEncoding, EncodingToString, or EncodingToEncoding.
TargetEncoding	Yes	Required for StringToEncoding and EncodingToEncoding modes. For the proper language-independent code parameter values, see *System Requirements and Supported Platforms* on Siebel SupportWeb.
SourceEncoding	No	Required for EncodingToString and EncodingToEncoding modes. For the proper language-independent code parameter values, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

NOTE: To ignore character conversion errors (invalid character errors or substitution errors) for the Convert method, specify IgnoreConversionErrors = TRUE as a method argument. Otherwise, do not use this argument. (This argument is hidden in Siebel Tools.)

About Application-Wide Data

This topic describes the different types of application-wide data that need to be considered for global deployments.

Sales Cycles

Sales cycle stages can be localized into different languages by configuring them according the global data approach described in "Setting Up Global Data" on page 52. See also Technical Note 438 on Siebel SupportWeb.

Periods

Periods appear in applications such as Incentive Compensation and cannot be localized.

Currency

Make sure that your Siebel Business Applications use the appropriate currency for your users. For information about currency administration, including exchange rates and currency conversion, see the *Applications Administration Guide*.

Telephone Numbers

You will need to specify telephone number formats for the countries in which you will be deploying. For information about how to perform this procedure, see the *Applications Administration Guide*.

Templates

There are many templates to consider when planning a global deployment. They include:

- Proposals
- Presentations
- Reports
- Email
- Fax
- Correspondence

  Each correspondence template is in a specific language. These templates have been localized. For information about creating new correspondence templates, in any language, see the *Siebel Correspondence, Proposals, and Presentations Guide*. Each user can specify a default correspondence template; users can choose the appropriate template for their locale.

  NOTE: You can have the language tagged in the template in the appropriate language or locale field. Because the Application Object Manager is multilingual, you can write a single workflow rule, then change the search specification on the language or locale of that template.

Literature

The items that are included in the Literature tab can be in any language. For information about adding items to the Literature tab, see the *Applications Administration Guide*.

NOTE: In order to be able to tell the language in which a literature item is written, you may want to include the item's language in its name and description, or add extension columns to hold those properties, so that they are searchable and could be automatically assigned by Assignment Manager.

iHelp

iHelp is a language-independent feature. This means that:

- The same iHelp can run in multiple languages.
- The same logic can be used, but with a localized message displayed.
- The language is determined by the current application language.

For more information about creating or administering iHelp, see the *Applications Administration Guide*.

Siebel SmartScript

Siebel SmartScript is a language-independent module. This means that:

- The same SmartScript can run in multiple languages.
- The same logic can be used, but with a localized message displayed.
- The language can be manually selected or set automatically when a script is started.

NOTE: SmartScript does not recognize date-time data as being in UTC format. It can recognize only date-time data as specified by the server it is running on.

For more information about SmartScript, see the *Siebel SmartScript Administration Guide*.

Setting Up Global Data

Global data is master data that is configured to display in the current application language. For example, a Call Center user accessing an FRA application would see data in French, while another user accessing a DEU application would see the same data in German.

To set up global data for your global deployment, follow steps similar to those in the example tasks in this section.

Example for Enabling Global Data

The following sample task describes how to use the Opportunity Product business component to enable the Product Description field as global data.

To enable global data (example)

1. In Siebel Tools, check out and lock the Opportunity Product project.
2. In the Object Explorer, navigate to Business Component.
3. In the Name field, perform a query for Opportunity Product.
4. In the Object Explorer, navigate to Join and add the following record:

Field	Value
Table	S_PROD_INT_LANG
Alias	S_PROD_INT_LANG
Outer Join	Yes

5 In the Object Explorer, select Join > Join Specification and add the following record:

Field	Value
Name	Translate
Destination Column	PAR_ROW_ID
Source Field	Product Id

6 In the Object Explorer, select Join > Join Constraint and add the following record:

Field	Value
Name	Language
Destination Column	LANG_ID
Value	One of the following values: ■ Language() corresponds to the value of the Language Code parameter. ■ LOVLanguage() is the language currently used for LOVs. It may correspond to the value of the Language Code parameter or to the Preferred Language setting for the current user.

NOTE: LOVLanguage() is designed to work with the multilingual outbound communication functionality. Language() does not work with this functionality.

7 In the Object Explorer, navigate to Field.

8 In the Name field, perform a query for Product Description.

9 Change the Join to S_PROD_INT_LANG and the Column mapping to DESC_TEXT.

Translating Globally Enabled Fields

The following sample task describes how to translate a globally enabled field, using the Product Description field from the preceding sample procedure as an example.

To translate a globally enabled field

1 Expose the view that contains the field you want to translate.

NOTE: In the example provided previously in this section, the view would already be exposed since Products is an object that is global-data ready.

2 Navigate to Administration - Product > Products view.

3 In the Products list, select a product that you want to translate.

4 In the More Info form, on the link bar, click Translations.

5 Enter the translated names and language codes for the product.

Configuring the Calendar

The Calendar screen in your Siebel applications can be configured to support the requirements of users based in different locales in a global deployment.

For example, the first day of the week displayed in the weekly and monthly calendar views can be specified using the applet user property First Day of Week. Integer values correspond to days of the week: 0 = Sunday, 1 = Monday, and so on. First Day of Week is set to 1 (Monday) by default for the applet HI Calendar Base Applet. You can set this user property differently for other locale-specific calendar applets, according to your user requirements. Multiple locale-specific calendar applets can be included in one view, and hidden or exposed as appropriate for each user, using personalization.

For more information about configuring the Calendar, see the *Applications Administration Guide*. See also *Using Siebel Tools*, *Configuring Siebel Business Applications*, and related documents.

Alternatively, an integration using Siebel Server Sync for Microsoft Exchange Server (SSSE) provides the Embedded Outlook Calendar deployment option, which may meet your global deployment requirements for calendar display. For more information, see the *Siebel Server Sync Guide*.

Configuring Cascading Style Sheets to Specify Different Fonts

To successfully run Siebel Business Applications, your cascading style sheet files (CSS files) for Siebel applications must specify fonts that are available on your systems and that support the languages and locales in which you will run the applications.

By default, the style sheets specify fonts for Windows platforms only. The specified fonts are not Unicode fonts, and may not support the character sets you require for all languages you may deploy. (The term *Unicode fonts* is used here loosely to refer to any font that contains a wide range of the characters required to support the many languages included in the Unicode standard.)

You can use utilities on your client systems, such as the Character Map utility on Windows, to determine which characters are available in any installed font.

For some example font settings that may be suitable on different operating system platforms for supported languages or locales, see Table 8 on page 55.

Some scenarios in which you may need to modify font specifications in your cascading style sheets include the following:

- If your database uses Unicode, and some users need to be able to view data using multiple languages, then the cascading style sheets applicable to these users may need to be modified to specify Unicode fonts.

  For example, if U.S. English (ENU) users need to view data using Japanese (JPN) characters, then Unicode fonts would be required for the ENU users, such as to change from Arial to Arial Unicode MS. If the JPN users do not require viewing data in other languages, then a Shift-JIS font may be specified for these users, because the display characteristics may be more suitable.

  NOTE: Style sheet files can include font names that contain non-ASCII characters only if the file is saved using UTF-8 encoding.

- If you are deploying your Siebel applications with any language that is not supported by the default fonts specified in the cascading style sheets, then the style sheets applicable to users for this language must be modified to specify a suitable font. See example fonts listed in Table 8 on page 55.
- If you are deploying Siebel applications on client machines running UNIX (using standard interactivity), your style sheets must specify fonts that are available on your UNIX systems. See example fonts listed in Table 8 on page 55.
- You may also need to specify different fonts in order to display special symbols such as mathematical symbols, to change the font to suit your company's design standards, and so on. (These scenarios are not specifically related to deploying languages.)

You may need to modify multiple cascading style sheet files, depending on which Siebel products and features you are deploying. Cascading style sheet files include main.css (for employee applications, which use high interactivity), dCCmain.css (for customer applications, which use standard interactivity), jctrls.css (also high interactivity), and several others.

NOTE: For Siebel applications that use high interactivity mode, the first font specified in each applicable cascading style sheet must be valid. Additional font names specified in a list are not used.

Cascading style sheet files and other types of files, such as image files, are copied from the Siebel Server to the Web server each time the Web server is restarted. Alternatively, administrators can update these files explicitly, as described in the *Siebel Installation Guide* for the operating system you are using.

For more information about modifying cascading style sheets, see *Configuring Siebel Business Applications* and the *Siebel Developer's Reference*. For more information about deploying cascading style sheet files, see *Going Live with Siebel Business Applications*. See also FAQ 2104 on Siebel SupportWeb.

Table 8 on page 55 lists some example font settings that may be suitable on different operating system platforms for supported languages or locales. For multilingual Unicode deployments, it may be suitable to specify Arial Unicode MS in place of Arial, in order to be able to display data in different languages.

Table 8. Example Fonts for Style Sheets for Supported Languages or Locales

Language or Locale	Siebel Language Code	Windows Font Names	UNIX Font Names		
			Solaris	AIX	HP-UX
Arabic (Saudi Arabia)	ARA	Windows XP: Arial Other Windows: Traditional Arabic	Naskh MT	Times New Roman WT J	Naskh
Chinese (Simplified)	CHS	SimSun	Zycjksun	Times New Roman WT J	Zycjksun

Table 8. Example Fonts for Style Sheets for Supported Languages or Locales

Language or Locale	Siebel Language Code	Windows Font Names	UNIX Font Names		
			Solaris	AIX	HP-UX
Chinese (Traditional)	CHT	MingLiu	Zycjksun	Times New Roman WT J	ARMingtiL
Czech	CSY	Arial	Arial	Times New Roman WT J	Arial
Danish	DAN	Arial	Arial	Times New Roman WT J	Arial
Dutch	NLD	Arial	Arial	Times New Roman WT J	Arial
English (United States)	ENU	Arial	Arial	Times New Roman WT J	Arial
Finnish	FIN	Arial	Arial	Times New Roman WT J	Arial
French (France)	FRA	Arial	Arial	Times New Roman WT J	Arial
German (Germany)	DEU	Arial	Arial	Times New Roman WT J	Arial
Hebrew	HEB	Windows XP: Arial Other Windows: Arial Hebrew	Arial Hebrew	Times New Roman WT J	Arial
Italian	ITA	Arial	Arial	Times New Roman WT J	Arial
Japanese	JPN	MS PGothic	(Font name uses Japanese characters)	Times New Roman WT J	(Font name uses Japanese characters)
Korean	KOR	MS Gulim	Gothic	Times New Roman WT J	Hybatang
Portuguese (Brazil)	PTB	Arial	Arial	Times New Roman WT J	Arial
Portuguese (Portugal)	PTG	Arial	Arial	Times New Roman WT J	Arial
Spanish (Modern)	ESN	Arial	Arial	Times New Roman WT J	Arial

Table 8. Example Fonts for Style Sheets for Supported Languages or Locales

Language or Locale	Siebel Language Code	Windows Font Names	UNIX Font Names		
			Solaris	AIX	HP-UX
Swedish	SVE	Arial	Arial	Times New Roman WT J	Arial
Thai	THA	Tahoma	AngsanaUPC	Times New Roman WT J	Angsa

Configuring Bidirectional Capability

By default, Siebel applications display user interface elements from left-to-right. To use Siebel Business Applications with Arabic (ARA) or Hebrew (HEB), you must perform various steps to change the user interface directionality.

Verifying UI Directionality Settings

Verify that UI Directionality is set to Right To Left for the Arabic and Hebrew locales.

- For the Siebel Web Client, the UI Directionality setting is already correctly defined by default as Right To Left for the Arabic and Hebrew locales, as shown in the view Administration - Data > Locales. Do not modify this setting.
- For Mobile and Developer Web Clients, the setting must be made directly in the application configuration files for ARA or HEB. To do this, add the following to the appropriate CFG file for your Siebel application, such as uagent.cfg for Siebel Call Center:

```
[InfraObjMgr]
UIDirectionality = RTL
```

Modifying Directionality in Siebel Web Templates

Framesets in Siebel Web templates (SWT files) need to be reversed on any Siebel Server that supports RTL displays. For more information about how to accomplish this, see Technical Note 599 on Siebel SupportWeb.

Modifying Fonts in Cascading Style Sheets

Depending on which fonts are available on your client machines, you may need to modify cascading style sheets (CSS files) to specify different fonts for use with your language. For more information, see “Configuring Cascading Style Sheets to Specify Different Fonts” on page 54. See also Technical Note 599 on Siebel SupportWeb.

Running Srvrcfg Command to Create Language-Specific Object Manager Components

If you are implementing a language that shipped after the initial product rollout, also run the `srvrcfg` command, as described in "Deploying Language Packs Shipped After Initial Rollout" on page 35.

5 Deploying with Global Time Zone

This chapter discusses how to deploy your Siebel Business Applications using the global time zone feature, for both new deployments and upgrades. It includes the following topics:

About Global Time Zone and Universal Time Coordinated (UTC)

Global deployments typically span multiple countries and have users working in several different time zones. The global time zone feature allows you to track dates and times consistently across time zones by specifying that you will use Universal Time Coordinated (UTC). Using UTC is strongly recommended. UTC is enabled by default for Siebel applications.

NOTE: UTC is essentially the equivalent of Greenwich Mean Time (GMT). However, certain localities using GMT or British Standard Time (BST) as a time zone observe daylight savings time (DST). DST does not apply for UTC itself, because it serves as a baseline against which all other time zone and DST offset adjustments apply.

The global time zone feature allows you to store date-time data in a common format and provides the following key benefits:

- Users can view dates and times in their local time zone, regardless of where work items were created.
- Users in different time zones can jointly handle team-based activities, such as managing service requests.

NOTE: Windows regional settings on client machines do not affect UTC. Although you can deploy your Siebel applications using UTC without setting the database server to UTC, you can optionally do so, as described later in this chapter.

CAUTION: If you do not want to deploy with UTC or are not ready to do so, you must review "Guidelines for UTC and Non-UTC Deployments" on page 62 and related topics.

Subtopics in this section are:

About UTC System Preferences

The global time zone feature makes use of two UTC-related system preferences:

- **Universal Time Coordinated.** Set to TRUE for UTC deployments (and set to TRUE by default). As described in "Converting Historical Data to UTC" on page 65, the UTC conversion utility sets this system preference to TRUE as its last step after completing the UTC conversion process. You do not need to set this system preference directly.
- **Default Time Zone.** Specifies a time zone that is to be used when the Time Zone field in the current user's profile is NULL, as described in "About UTC Data Conversion" on page 60. This system preference has no default value.

  For example, for a company with the majority of employees in California, you might set this system preference to (GMT-08:00) Pacific Time (US & Canada); Tijuana. The value you enter must correspond to the name of the time zone as represented in the Time Zone Administration view. For more information, see "Administering Time Zones" on page 75.

  NOTE: If you deploy with UTC, you must specify a value for Default Time Zone. Otherwise, incorrect time stamp values will be created and displayed in any case where no time zone preference is specified for an individual given user.

  For information about setting this system preference, see "Setting UTC System Preferences" on page 63.

NOTE: If you are *not* using UTC and the Universal Time Coordinated system preference is FALSE, the time stamps for records use the local time set for the operating system clock of the Siebel Server machine, which can use any time zone. The Default Time Zone system preference is never used if Universal Time Coordinated is FALSE.

About UTC Data Conversion

The global time zone feature converts date-time data to and from UTC. When Siebel applications are deployed using UTC, some date-time fields store data as UTC data.

To support UTC conversion, applicable date-time field object definitions use the data type DTYPE_UTCDATETIME (instead of DTYPE_DATETIME), and applicable column object definitions use the physical type UTC Date Time (instead of Date Time). UTC conversion code is invoked by the Application Object Manager in user sessions where applicable, as described below.

NOTE: To be subject to UTC conversion, custom date-time fields you create must be date-time fields *only*, not date-only or time-only fields, and must use the types mentioned above. See also "Enabling Custom Date-time Fields and Columns for UTC" on page 65.

There are some exceptions where date-time fields are not enabled for UTC. For more information, contact Siebel Technical Support.

Where the Universal Time Coordinated system preference is TRUE, UTC conversion in runtime operations occurs as described below:

- During runtime operation, date-time data that is entered by users (for example, appointment start or end times) is converted to UTC based on the time zone that is stored in the user's profile. If the Time Zone field in the user's profile is NULL, the time zone defined by the Default Time Zone system preference is used for this purpose instead. Daylight savings adjustments take effect according to the calendar date for the entered data.
- Date-time data that is displayed to users is converted *from* UTC based on the time zone that is stored in the user profile, including any daylight savings adjustments. If the Time Zone field in the user's profile is NULL, the time zone defined by the Default Time Zone system preference (with daylight savings adjustments) is used for this purpose instead.
- Date-time data that is entered into the database by the system (date-time stamp) is converted to UTC by the Application Object Manager before it is stored, based on the time zone of the application server machine.
- Siebel components that are not Object Manager-based (for example, Workflow Policies or Siebel EIM components on the Siebel Server) invoke UTC-specific database functions in order to handle date-time data consistently for UTC deployments.

See also "About UTC System Preferences" on page 60.

Example of UTC Data Conversion

To illustrate how UTC data conversion works, suppose a scenario where User 1 is in New York, User 2 is in San Francisco, and the database server and the Siebel Server are in Utah. Today's date is November 20th. The time zone for User 1 is equivalent to UTC - 5 (Eastern Standard Time), and the time zone for User 2 is equivalent to UTC - 8 (Pacific Standard Time).

User 1 logs in and books a meeting with User 2. User 1 sets the meeting for December 15, 1 pm New York time, which is UTC - 5. The time stored in the database is changed by the Application Object Manager (AOM) component on the Siebel Server before storage by applying this user's time zone offset, UTC - 5, to the actual time value. So the meeting time stored is 1 pm (EST) + 5 hours, which is 6 pm UTC (1800 in 24-hour clock time, which is what is actually stored).

Note that because the meeting date is in December, and because the date when the users are viewing the meeting is in November, no adjustments for daylight savings time (DST) are necessary.

User 2 now looks at her calendar. She sees the time for the meeting as 10 am. This is because the AOM reads the value of 1800 from the database, and applies her time zone offset to it, UTC - 8. 1800 - 8 hours is 1000, or 10 am (PST).

Guidelines for UTC and Non-UTC Deployments

This topic describes some guidelines that apply for deploying with UTC, or for deploying without UTC. Additional requirements are noted in other topics to which they apply.

CAUTION: Once you have already deployed your Siebel applications using UTC and created user date-time data in UTC format, *it is not supported to stop using UTC* by setting the Universal Time Coordinated system preference to FALSE. Otherwise, incorrect time stamp values may be created and displayed.

If you do not want to deploy with UTC or are not ready to do so, it is possible to set the Universal Time Coordinated system preference to FALSE in limited circumstances:

- For a new installation, you can set Universal Time Coordinated to FALSE after completing installation steps, *as long as you have not created any user date-time data in UTC format*.
- For an upgrade to version 7.8.x from a prior Siebel version deployed without UTC, Universal Time Coordinated may be either FALSE or TRUE (depending on the version you upgrade from). You can set this system preference to FALSE, *as long as you have not created any user date-time data in UTC format, and have not converted historical date-time data to UTC format*.

For more information, see "About UTC System Preferences" on page 60 and "Setting UTC System Preferences" on page 63.

CAUTION: For an upgrade to 7.8.x, if Universal Time Coordinated is TRUE following the upgrade and you do intend to deploy with UTC, you *must* run the UTC conversion utility before any new date-time data is created in UTC format. For details, see "Converting Historical Data to UTC" on page 65.

For an upgrade to version 7.8.x that you will deploy with UTC, you perform all UTC conversion steps within the overall context of upgrade and roll-out processes described in *Siebel Bookshelf* documentation. For example, for deployments using Siebel Replication Manager, all regional nodes must be synchronized prior to upgrading, then re-extracted after upgrading and converting to UTC. For more information, see *Going Live with Siebel Business Applications*, the *Upgrade Guide*, and the *Siebel Remote and Replication Manager Administration Guide*.

If you deploy with UTC for new or upgrade deployments, you must set the Default Time Zone system preference. See "About UTC System Preferences" on page 60 and "Setting UTC System Preferences" on page 63.

When deploying with UTC for new or upgrade deployments that contain custom date-time fields, you must perform the tasks in "Enabling Custom Date-time Fields and Columns for UTC" on page 65.

Deploying Siebel applications using UTC does not require you to set the operating system clock for the database server machine to UTC (a GMT time zone without DST). This optional task is described in "Setting the Database Server to UTC" on page 64.

Siebel eScript provides a set of methods for working with UTC data, along with methods for non-UTC date-time data. Use the appropriate methods for your deployment. All processing of UTC data performed by eScripts is completely independent of the UTC conversion capabilities of the Application Object Manager. For information about the UTC methods in eScript, see the *Siebel eScript Language Reference*.

If you import appointment or activity data into the Siebel application (with UTC on) from another system that does not use UTC, additional requirements apply. For more information, see "Converting Historical Data to UTC" on page 65.

About Enabling UTC for Existing Deployments

If you are upgrading a non-UTC deployment of a prior version and are now deploying with UTC, or if you have decided to convert an existing non-UTC implementation to UTC, you must perform applicable tasks described in the following topics:

- "Setting UTC System Preferences" on page 63
- "Setting the Database Server to UTC" on page 64 (this task is optional)
- "Enabling Custom Date-time Fields and Columns for UTC" on page 65 (this task also applies whenever you create custom date-time fields)
- "Converting Historical Data to UTC" on page 65

See also "Administering Time Zones" on page 75.

Setting UTC System Preferences

UTC deployments use two system preferences, as described in "About UTC System Preferences" on page 60.

Generally you do not need to set the Universal Time Coordinated system preference, because it is set to TRUE by default for new deployments, and the UTC conversion utility also sets it to TRUE when it completes the conversion process.

However, for a UTC deployment, you must set the Default Time Zone system preference to the time zone that is to be used when the Time Zone field in the current user's profile is NULL. Complete the procedure in this topic to set this system preference.

For more information about upgrading historical data to UTC (in an upgrade or non-upgrade environment), see "Converting Historical Data to UTC" on page 65.

CAUTION: If you do not want to deploy with UTC or are not ready to do so, you must review "Guidelines for UTC and Non-UTC Deployments" on page 62 and related topics.

To set the UTC system preferences

1. Navigate to Administration - Application > System Preferences.
2. Locate the Universal Time Coordinated system preference and review the value in the System Preference Value field. The default value is TRUE.
   - If you are deploying with UTC, do not change the value. (Even if the value is FALSE, as for an existing non-UTC deployment, you do not need to change it, because the UTC conversion utility sets it to TRUE automatically.)
   - If you are deploying without UTC, you can change the value to FALSE under limited circumstances, as outlined in "Guidelines for UTC and Non-UTC Deployments" on page 62.

3 Locate the Default Time Zone system preference. In the System Preference Value field, enter your preferred default time zone.

For example, for a company with the majority of employees in California, you might set this system preference to (GMT-08:00) Pacific Time (US & Canada); Tijuana. The value you enter must correspond to the name of the time zone as represented in the Time Zone Administration view. For more information, see "Administering Time Zones" on page 75.

Setting the Database Server to UTC

As part of enabling UTC for a new or upgrade deployment, you can optionally set the operating system time of the database server for the Siebel Database to a time zone based on GMT.

NOTE: Deploying Siebel applications using UTC does not require you to set the operating system clock for the database server machine to UTC (a GMT time zone without DST), as described here. This task is optional. See also "Guidelines for UTC and Non-UTC Deployments" on page 62.

The specific method of setting the system time to UTC on the database server depends on the operating system you are using. Many operating systems have a time zone setting called UTC or something similar. Other systems have time zone options described as Greenwich Mean Time without daylight savings time—the equivalent of UTC.

NOTE: Some databases, such as Oracle, provide listener programs which can be run on other machines than the database server machine. If you set the system clock on the database server machine to UTC, it is recommended to also set the time on these listener machines to UTC.

To set the database server machine to use UTC in Windows

1 In the Windows task bar, double-click the time indicator to display the Date and Time Properties dialog box.

2 Click the Time Zone tab.

3 Specify (GMT) Casablanca, Monrovia.

Select this GMT time zone for UTC because daylight savings time (DST) is not observed. For more information, see "About Global Time Zone and Universal Time Coordinated (UTC)" on page 59.

4 Turn off the option Automatically adjust clock for daylight saving changes.

NOTE: This step is not strictly required if you chose (GMT) Casablanca, Monrovia instead of a different GMT time zone, but may be considered precautionary.

5 Click OK.

To set the database server machine to use UTC in UNIX

- Set TZ=UTC0 in the /etc/config file.

Use the system-level time zone unless you need to have a different time zone for a particular purpose. In this case, set the time zone at the shell level by following the vendor documentation for your operating system.

Enabling Custom Date-time Fields and Columns for UTC

Some date-time fields (and their underlying columns) in Siebel Business Applications are enabled for UTC. This includes fields that you manually populate by entering date-time data and fields that the system populates by generating a date-time stamp. If you create custom date-time fields and columns that will store UTC data, you must enable them for UTC, so that data entered in these fields is consistent with data entered in other UTC-enabled date-time fields.

See also "About UTC Data Conversion" on page 60 and related topics in "About Global Time Zone and Universal Time Coordinated (UTC)" on page 59.

If you configure additional date-time columns for UTC, you need to add these columns to the file utc_columns.inp. This input file tells the UTC conversion utility which columns need to be converted to UTC. Do this before you run the UTC conversion utility, which is described in "Converting Historical Data to UTC" on page 65.

For more information about configuring column and field object definitions, see *Using Siebel Tools*.

To enable custom date-time fields for UTC

- In Siebel Tools, set properties for each applicable column object definition and its associated field object definition:
  - a For the column, set the Physical Type property to UTC Date Time.
  - b For the field, set the Type property to DTYPE_UTCDATETIME.

  If the Universal Time Coordinated system preference is set to TRUE, after you compile your changes these custom date-time fields will be enabled for UTC.

Converting Historical Data to UTC

When you enable the global time zone feature, you need to convert any historical (non-UTC) time data to UTC. Perform applicable tasks in this section if you are upgrading from a Siebel deployment of a prior version that did not use UTC, or if you have decided to convert an existing non-UTC implementation to UTC.

See also "Guidelines for UTC and Non-UTC Deployments" on page 62.

Converting historical data helps ensure that existing date-time values are consistent with the global time zone logic. The global time zone feature stores date-time values in the database adjusted to UTC time. If you do not convert it, your historical data will be incorrect for a UTC deployment.

For more information about how the global time zone feature processes date-time data in UTC, see "About UTC Data Conversion" on page 60.

Although enabling UTC is optional, it is recommended that you perform the following procedures immediately after upgrading to the current Siebel release. If you have upgraded a non-UTC deployment, you *must* perform the conversion steps described here before creating any new UTC date-time data—unless you have set the Universal Time Coordinated system preference to FALSE. For more information, see "Guidelines for UTC and Non-UTC Deployments" on page 62.

The tasks in this section run the UTC conversion utility from the Database Server Configuration Wizard. For more information about running the Database Server Configuration Wizard, see the *Siebel Installation Guide* for the operating system you are using and see the *Upgrade Guide*.

Before beginning to convert historical data to UTC, you must upgrade your development and production environments to the current Siebel release. For more information, see the *Upgrade Guide*. For upgrades, you may also need to convert UTC delta columns, for columns that were not UTC-enabled in prior releases. For more information, see "Performing UTC Delta Conversion" on page 75.

Before you run the UTC conversion utility, you should drop all database triggers. You can recreate or reenable the triggers after the UTC conversion is complete.

NOTE: If you are preparing external data, including appointments, for example, to be brought into a Siebel Database using Siebel EIM or Siebel EAI, you must modify the time stamp for each individual record to be consistent with UTC. For each record, the updated time stamp must be based on the applicable UTC offset. The offset must take into account both the time zone offset from UTC (for the appointment's location) and any applicable daylight savings time (DST) offset from the standard time zone that will be in effect (for the appointment's calendar date).

Process of Converting Historical Data to UTC

To convert historical data for use in an environment using UTC, perform the tasks below:

1. "About the UTC Conversion Utility" on page 66
2. "Preparing Your Data for Conversion to UTC" on page 68
3. "Running the UTC Conversion Utility" on page 72
4. "Reviewing the UTC Conversion Log Files" on page 74

You may also need to do the following:

- "Manually Launching the UTC Conversion Utility" on page 74
- "Performing UTC Delta Conversion" on page 75

About the UTC Conversion Utility

To update existing date and time data in your data tables to UTC, you need to run the UTC conversion utility from the Database Server Configuration Wizard, as described in "Running the UTC Conversion Utility" on page 72. This utility helps you define the required parameters for UTC conversion.

After conversion is complete, the UTC conversion utility sets the Universal Time Coordinated system preference to TRUE. (When you launch the utility, the setting can be either TRUE or FALSE.)

Running the UTC conversion utility reads a series of input files that control the conversion of your date-time data to UTC. Several input files are provided by default. The input files are located in the directory *SIEBEL_ROOT*\DBSRVR*DB_PLATFORM*, where *DB_PLATFORM* is DB2UDB, DB2390, MSSQL, or ORACLE. The input file names start with *utc* and have the extension *.inp*.

The file driver_utc.ucf identifies the UTC conversion input files, which contain parameters specifying the columns you are converting to UTC. Each of these input files updates the appropriate database columns to the UTC format in a single database transaction.

Each input file entry contains the table name, a WHERE clause, and a list of columns with their conversion methods. The conversion method defines how to link each record to the user record from which the default time zone is derived. The value specified as the conversion method corresponds to a column whose value identifies a unique user record.

NOTE: It is recommended to modify the default input files, or to create additional input files, to specify all applicable date-time columns to be converted, and to specify the most suitable conversion method for each column. The conversion methods are explained below.

Each input file entry is structured as follows:

```
[TABLE_NAME]
Clause = WHERE_CLAUSE
Column = COLUMN_NAME, CONVERSION_METHOD
...
```

where:

- *TABLE_NAME* is the database table containing the date-time columns you will be converting.
- *WHERE_CLAUSE*, an optional value for Clause, can be used to specify a subset of columns to be converted. For an example, see "Preparing Your Data for Conversion to UTC" on page 68.
- *COLUMN_NAME*, the first value for Column, indicates a column to be converted (for example, CREATED).
- *CONVERSION_METHOD*, the second value for Column, indicates the conversion method to be used for this column. Possible values are SERVER_TIME and CREATED_BY.

Conversion Methods Used by the UTC Conversion Utility

The two conversion methods used by the UTC conversion utility are described in greater detail below:

- **SERVER_TIME.** Conversion will be done according to the time zone of the server, as specified in the file server_time.inp. This method should be used for date and time values that are not associated with a user.

  You must use the correct syntax in the server_time.inp file. If the exact values are not specified, the UTC conversion utility will not work. Use the time zone setting for your operating system to determine the correct values for this syntax. Enter the value and region (in quotation marks), on the second line of the server_time.inp file.

  Here is an example for U.S. Central Standard Time (CST):

  ```
  [GLOBAL] SERVER_TIME: "(GMT-06:00) Central Time (US & Canada)";
  ```

- **CREATED_BY.** Conversion will be done according to the time zone of the user who created the record. The UTC conversion utility will look up the profile of the user who created the record being converted, and derive the corresponding time zone. In general, this conversion method is appropriate for converting data created by individual users, where creation time is relevant.

  An example of this type of data would be service requests originally created by users in multiple time zones, where for each record the time stamp before conversion is in the user's local time. In this case, you may choose to modify the input file s_srv_req.inp to use CREATED_BY instead of SERVER_TIME. Each record's time stamp will be converted to UTC based on the user's time zone.

Preparing Your Data for Conversion to UTC

Before you convert your data to UTC, you need to prepare it.

This task is a step in "Process of Converting Historical Data to UTC" on page 66.

- Configure custom fields and columns (if necessary) for UTC. For more information, see "Enabling Custom Date-time Fields and Columns for UTC" on page 65.
- Set time zones for each of your users.
- Save a report of your user time zones.
- Edit the driver_utc.ucf file to specify more input files.
- Modify the default input files as needed for your UTC deployment.
- Modify input files for partitioned tables, to assure sufficient log space.
- Allocate maximum database transaction log space.

NOTE: Log space is controlled through transaction logs. On DB2 UDB, this is called transaction log. On Oracle, this is called rollback segment. On MS SQL Server, this is called log file.

Set Time Zones for Users

For each of your users, specify the time zone in the Contacts screen. This data is stored in the S_CONTACT table.

You must also specify a value for the Default Time Zone system preference. For more information, see "Setting UTC System Preferences" on page 63.

Save a Report of User Time Zones

Prior to running the UTC conversion utility, save a report of your user time zones as a record of the input data used during the conversion.

Edit the driver_utc.ucf File to Specify More Input Files

The file driver_utc.ucf identifies the input files for UTC conversion. If you create additional input files, you need to add them to driver_utc.ucf. A default set of input files is provided. You may need to create additional files to specify more columns to convert.

An example from the driver_utc.ucf file appears as follows:

```
[File Execute Entry 7]

Type = FileExecute

File Name = $SiebelRoot\bin\utcupgd

Check Return Code = 1

Return Code Compliance = 0

16 Bit App = 0

Command Line = /u $UserName /p $Password /c "$ODBCDataSource" /d $DatabaseOwner /n
"$RepositoryName" /g $Language /x $DatabasePlatform /j $SiebelRoot/bin/
s_camp_con_01.inp /l $SiebelRoot/log/s_camp_con_01.log /s $SiebelRoot/bin/
server_time.inp

Number of 10 Second Wait Loops = 2000

Prompt User For Status = 0

Parallelizable Item = 0

Title Message Num = 0

Estimated Disk Space = 0

Backup Db = 0
```

To edit driver_utc.ucf file parameters to accommodate additional input files

1. Specify a new input file in the driver_utc.ucf file by copying a complete step from the driver_utc.ucf file and pasting the copied step immediately after the step you copied.
2. In the new step, change the input file parameter (which follows /j in the command line) to the name of your new input file.

   Using the preceding example in the driver_utc.ucf file, change the name of the input file in the new step from this:

   ```
   /j $SiebelRoot/bin/s_camp_con_01.inp
   ```

   to reflect the new input file name:

   ```
   /j $SiebelRoot/bin/new_file_name.inp
   ```

3 In the new step, change the log file parameter (which follows `/l` in the command line) to the name of the log file that corresponds to your new input file.

Using the preceding example in the driver_utc.ucf file, change the name of the log file in the new step from this:

```
/l $SiebelRoot/log/s_camp_con_01.log
```

to reflect the new log file name:

```
/l $SiebelRoot/log/new_file_name.log
```

4 Repeat Step 1 on page 69 through Step 3 on page 70 for each new input file.

5 Renumber the file execute entry numbers for your new step and for each subsequent step, in order.

Using the preceding example in the driver_utc.ucf file (and assuming no preceding steps have changed), you would change the execute entry number in the new step from this:

```
[File Execute Entry 7]
```

to reflect the next step in the sequence:

```
[File Execute Entry 8]
```

and so on, for each subsequent step.

Modify Default Input Files As Needed

Modify the default input files for the UTC conversion utility as appropriate for your deployment, or create additional input files. An example of customizing input files to partition data into separate input files is provided later in this section. See also "About the UTC Conversion Utility" on page 66.

Partitioning Prerequisite for Oracle

By default, some tables are partitioned by making use of month (in the CREATED column). Make sure that the user running the UTC conversion utility has execute privilege on month. The following task describes how to give the execute permission to users who do not have it.

To grant execute permission

1 Connect to the database server as the tableowner.

2 Execute the following command:

```
grant execute on month to SSE_ROLE
```

Modify Input Files for Partitioned Tables

As the UTC conversion utility processes input files, it typically processes each table as a whole. However, tables with very large record counts may encounter errors due to constraints on log space at the database level.

The UTC conversion utility prevents errors that could occur due to insufficient log space at the database level, by using multiple input files to partition large tables into subsets of records for processing. The utility updates each record set individually to convert all rows in a partitioned table.

The UTC conversion utility uses partition keys to control how a table is divided into record sets. For example, large tables may be divided based on the calendar month in which each record was created, resulting in twelve approximately equal-sized partitions.

Partition keys are supplied for tables that are typically very large and that generally use a lot of log space if updated as a single input file. The tables that are delivered with partition keys are:

- S_CAMP_CON
- S_COMMUNICATION
- S_EVT_ACT
- S_SRV_REQ
- S_ORG_EXT
- S_CONTACT

The default value for each key is customizable. You determine the way that your tables are partitioned, and you can partition your own tables that you know to have large record counts by adding or modifying the input files.

NOTE: If you require a different partitioning method, or if you want reduced partitioning in order to optimize performance, contact Siebel Technical Support or Expert Services for assistance. If you create additional partitioned files, you may decrease performance.

The input file includes a WHERE clause, which defines the parameters that will be used as partition keys to divide large tables into appropriately sized sections. This WHERE clause represents standard SQL that will be used to filter which records are to be updated by each input file. Verify that you are using the correct SQL syntax.

The following example is from the file s_evt_act_00.inp. This particular file is used to define one partition of the activities table that includes all records created in the month of January (month=1) or February (month=2).

```
[S_EVT_ACT]
Clause = where month(CREATED) = 1 or month(CREATED) = 2
Column = APPT_START_DT, CREATED_BY
Column = TODO_ACTL_END_DT, CREATED_BY
Column = TODO_ACTL_START_DT, CREATED_BY
Column = TODO_AFTER_DT, CREATED_BY
Column = TODO_DUE_DT, CREATED_BY
Column = TODO_PLAN_END_DT, CREATED_BY
Column = TODO_PLAN_START_DT, CREATED_BY
```

NOTE: If the WHERE clause is blank, then the table will not be partitioned, and will be processed as a whole.

Allocate Maximum Log Space

Prior to running the UTC conversion utility, set the log space parameters on the database server to the maximum. The utility requires a large amount of log space in order to run properly.

For Oracle, allocate a single large rollback segment and take other rollback segments offline to make sure that large transactions succeed.

Running the UTC Conversion Utility

Complete the following task to run the UTC conversion utility and convert your date-time data to UTC.

See also "About the UTC Conversion Utility" on page 66 and "Preparing Your Data for Conversion to UTC" on page 68.

This task is a step in "Process of Converting Historical Data to UTC" on page 66.

NOTE: Running the UTC conversion utility may take a long time, depending on how many records and how many columns are being converted.

To run the UTC conversion utility from the Database Server Configuration Wizard

1 Launch the Database Server Configuration Wizard by using one of the following methods:

- On Microsoft Windows, launch the utility by selecting Start > Programs > Siebel Enterprise Server Configuration 7.x > Database Server configuration.
- On UNIX, first set these environment variables:

  setenv `LANGUAGE` *`lang`* (where *`lang`* represents your display language, such as ENU)

  setenv `SIEBEL_ROOT` *`SIEBEL_ROOT`* (where *`SIEBEL_ROOT`* is the name of your Siebel root directory)

  Then type the following command to launch the utility:

  ```
  dbsrvr_config.ksh
  ```

The Database Server Configuration Wizard appears.

NOTE: The values you enter into the Database Server Configuration Wizard are case-sensitive. You may not type spaces in parameter values; use underscores (_) instead.

2 Enter the required parameters to run the UTC conversion utility.

See Table 9 on page 73 for a list of dialog boxes, options, and required values.

The UTC conversion utility updates your existing data. For columns configured for UTC, the UTC conversion utility adjusts the historical date and time values to their UTC equivalent.

After successfully converting the UTC-enabled date-time fields, the UTC conversion utility sets the value for the Universal Time Coordinated system preference to TRUE.

The Database Server Configuration Wizard checks for errors, and writes any errors to a log file.

3 Review the log file generated by the UTC conversion process, and resolve errors as necessary. See "Reviewing the UTC Conversion Log Files" on page 74.

NOTE: If the UTC conversion fails for any reason, you must review the log files and resolve any errors encountered. Then you need to manually rerun the UTC conversion utility, as described in "Manually Launching the UTC Conversion Utility" on page 74.

Table 9 on page 73 shows parameters to use with the Database Server Configuration Wizard.

Table 9. Database Server Configuration Wizard

At this prompt...	Enter or select the following...
Siebel Enterprise Parameters: Siebel Gateway Name Server Address	Siebel Gateway Name Server Address (This is typically the machine name.) Enterprise Server Address
Installation and Configuration Parameters: Siebel Server Directory	Siebel Server Directory (This is the *SIEBSRVR_ROOT* directory, for example, D:\sea7x\siebsrvr.)
Installation and Configuration Parameters: Siebel Database Server Directory	Database Server Directory (This is the *DBSRVR_ROOT* directory, for example, D:\sea7x\dbsrvr.)
Database Server Options: Siebel Database Operation	Run Database Utilities
Database Utilities: Database Utility Selection	Universal Time Code Conversion
Installation and Configuration Parameters: Language Selection	Base language of your Siebel application (This is the primary language of your prior environment.)
Installation and Configuration Parameters: RDBMS Platform	RDBMS Platform
Installation and Configuration Parameters: ODBC Data Source Name	ODBC data source[1] (This was created automatically by the Siebel Server installation, using the format *SiebSrvr_EnterpriseName*.)
Installation and Configuration Parameters: Database User Name	Database User Name (User name of the Siebel administrator, for example, sadmin) Database Password
Installation and Configuration Parameters: Table Owner	Table Owner Name (Note for Microsoft SQL: this is the login for the owner of the database, not necessarily the default owner of the database in DBO.) Table Owner Password
UTC Parameters: Repository Name	Repository Name
Configuration Parameter Review	Review the parameters you have defined and then click Finish.

1. To find the name of the ODBC data source on Windows XP, go to Start > Settings > Control Panel > Administrative Tools > Data Sources (ODBC). Click the System DSN tab to find the name of the ODBC data source.

Reviewing the UTC Conversion Log Files

After the UTC conversion utility runs, if errors are encountered, the utility records those errors to log files. The log files are located in the log subdirectory of *SIEBSRVR_ROOT*.

This task is a step in "Process of Converting Historical Data to UTC" on page 66.

Carefully review the log files for errors. Some of the errors listed in the log file are acceptable, but others will require resolution.

An example of an acceptable error is when the utc_drop_temp_tab.log file generates the error "ORA-00942: table or view does not exist" when initially dropping the S_CONTACT_TMP and S_TIMEZONE_TMP tables.

The default log files are listed below. You may have additional log files if you modified the input files.

```
utc_drop_temp_tab.log
null_timezone.log
utc_insert_to_tmp_tab.log
utc_create_tmp_ind.log
utc_run_stats.log
s_camp_con_00.log
s_camp_con_01.log
s_camp_con_02.log
s_camp_con_03.log
s_communication_00.log
s_communication_01.log
s_communication_02.log
s_communication_03.log
s_evt_act_00.log
s_evt_act_01.log
s_evt_act_02.log
s_evt_act_03.log
s_evt_act_04.log
s_evt_act_05.log
s_contact.log
s_org_ext.log
s_srv_req.log
utc_columns.log
denorm.log
utc_drop_temp_tab2.log
```

Review the log files that were generated by the UTC conversion utility, including any custom log files that you may have created, and resolve errors as necessary. If the UTC conversion utility was interrupted after it encountered an error, you need to manually relaunch the UTC conversion utility.

Manually Launching the UTC Conversion Utility

If UTC conversion fails for any reason, you need to review the log files and resolve any errors encountered. Then you need to manually rerun the UTC conversion utility.

To manually launch the UTC conversion utility

- Type the following command from the bin subdirectory in *SIEBSRVR_ROOT*:
  - In a Windows DOS prompt, type: `siebupg.exe /m ..\..\dbsrvr\`*`DB_PLATFORM`*`\master_utc.ucf`
  - In a UNIX shell, type: `srvrupgwiz /m ../../dbsrvr/`*`DB_PLATFORM`*`/master_utc.ucf`

  where *DB_PLATFORM* is DB2UDB, DB2390, MSSQL, or ORACLE.

Performing UTC Delta Conversion

You may want to perform a UTC delta conversion if you have upgraded to Siebel Business Applications, Release 7.0.x, performed the UTC conversion, and then upgraded to the current version of Siebel Business Applications. Completing this task converts the data in columns that changed from non-UTC to UTC in the Siebel Repository.

To perform the UTC delta conversion

1. Open *SIEBEL_ROOT*\DBSRVR*DB_PLATFORM*\master_utc.ucf, where *DB_PLATFORM* is DB2UDB, DB2390, MSSQL, or ORACLE.
2. In master_utc.ucf, replace all occurrences of *driver_utc.ucf* with *driver_utc_delta.ucf*.
3. Launch the Database Server Configuration Utility to run the UTC conversion utility, as described in "Running the UTC Conversion Utility" on page 72.

Administering Time Zones

The time zone records that are shipped with Siebel Business Applications as seed data include the 74 world time zones. These are the same time zones used by Microsoft in its operating systems. Time zone records include the time zone name and standard abbreviation, its offset from UTC, daylight savings time (DST) data, and so on.

Procedures are provided below for modifying time zones and for maintaining translations of time zones.

For more information about managing time zone data, see also the *Applications Administration Guide*.

Modifying Time Zones

You will rarely need to modify a time zone's regional settings. However, you may want to modify other time zone settings such as the display name or daylight savings rules, as these occasionally change.

To modify time zone settings

1. Navigate to Administration - Data > Time Zone Administration.

2 In the Time Zones list, select the time zone record you want to modify.

3 Modify the fields as necessary. Fields are described in the following table:

Field	Description
Name	Full name of time zone. **NOTE:** For UTC deployments, you specify the full name of the applicable time zone as the value for the Default Time Zone system preference. For more information, see "About UTC System Preferences" on page 60.
Standard Abbreviation	The abbreviation for the time zone.
UTC Offset	The time difference in minutes between local time and UTC. For example, the (U.S.) Eastern Standard Time has a UTC offset of -300 (GMT-5). Offsets are not necessarily in increments of hours. For example, Adelaide (Australia) has a UTC offset of 570 (GMT+9:30).
DST Abbreviation	The abbreviation for daylight savings time for the time zone.
DST Bias	The difference in time, in minutes, DST makes (where applicable). For example, the (U.S.) Eastern Standard Time (GMT-5) has a DST bias of 60.
DST Start Ordinal	Part of the rule that determines when DST starts. For example, if the rule is *the first Sunday in April*, then *First* is defined in this field.
DST Start Day	Part of the rule that determines when DST starts. For example, if the rule is *the first Sunday in April*, then *Sunday* is defined in this field.
DST Start Month	Part of the rule that determines when DST starts. For example, if the rule is *the first Sunday in April*, then *April* is defined in this field.
DST Start Time	Start time for DST, measured in minutes into the day when DST starts. For example, 2 a.m., a common DST start time, is equivalent to 120 minutes.
DST End Ordinal	Part of the rule that determines when DST ends. For example, if the rule is *the last Sunday in October*, then *Last* is defined in this field.
DST End Day	Part of the rule that determines when DST ends.
DST End Month	Part of the rule that determines when DST ends.
DST End Time	End time for DST, measured in minutes into the day when DST ends.
Active	Specifies whether a time zone record is considered active (when Active = TRUE).

Administering Time Zone Translations

You can view and maintain translations of the text-based fields for each time zone using the Time Zone Administration view in the Administration - Data screen.

To administer translations of time zones

1. Navigate to Administration - Data > Time Zone Administration.
2. In the Time Zones list, select the time zone for which you want to view translations, or that you want to translate.
3. The Time Zone Translations list displays existing translations.
4. If you require a new translation, add a new record and enter the translated version of the current time zone.

6 Localizing Global Deployments

This chapter describes how to localize your customized Siebel application for a global deployment. It includes the following topics:

About the Localization Process

Localizing a Siebel application includes translating the user interface and modifying other attributes to meet locale-specific requirements.

NOTE: It is recommended that you complete as much configuration as possible *before* you begin localizing the application. One reason for doing so is that configuration potentially alters what needs to be translated, so it is better to have a stable application. Also, the check-in and check-out mechanism in Siebel Tools is meant to be performed in only one language; if you perform localization on the client during configuration, there is a potential for data loss.

For detailed information about working with user interface strings (including symbolic strings) and other locale-specific data, see *Using Siebel Tools*. You can use the Locale Management Utility (LMU) in Siebel Tools to manage strings and other data. More information about the LMU is provided later in this section.

For information about localizing strings for Lists of Values (LOVs) and multilingual lists of values (MLOVs), see "Localizing Lists of Values and Multilingual Lists of Values" on page 84.

Localization Process

The localization process typically includes the following steps:

1. Identify the applications or projects that you want to localize.
2. Develop a localization glossary.

3 Export strings and other localizable attributes using the Locale Management Utility.

This utility exports the objects to a flat file. See LMU export limitations, later in this section.

When exporting symbolic strings, export the projects containing the symbolic strings, such as the Symbolic Strings project or a project you created that contains symbolic strings. For guidelines, see *Using Siebel Tools*.

NOTE: If you are localizing strings for an unshipped language (such as ENG, British English), you might export strings from a source language that was shipped by Siebel Systems (such as ENU, U.S. English), and specify your unshipped language as the target language (such as ENG). In the export file, copy the text from the source language to the target language. (Where many strings are similar or identical between the source and target languages, as with ENG and ENU, copying strings will speed up the translation process.) See also "Localizing an Unshipped Language" on page 85.

4 Translate strings by modifying the flat file directly or by importing the file into a separate localization development environment, modifying the locale-specific attributes, and then exporting the localization result to another flat file.

5 Import modified string and modified object definitions into the repository using the Locale Management Utility.

6 If necessary, search for strings or locale-specific attributes that have been modified since the last export, and update the string translation or attributes localization for these changed objects.

7 With the correct language mode specified in Siebel Tools, compile the modified projects into a repository file (SRF).

8 Distribute the SRF file to the appropriate Siebel Servers and clients.

Limitations to the LMU Export Process

The Locale Management Utility does not export menu items, nor all applets from an application for localization. The following applets are not exported with the LMU utility:

- SWE Export Applet
- SWE Import Applet
- Technical Support Applet
- About Record Applet
- About SRF Applet
- About Siebel Applet
- About View Applet
- Parametric Search Form Applet
- Delete Query Applet
- Save Query As Applet
- Change Records Popup Applet (SWE)

To perform a complete localization of one application, the following projects must be exported along with the application by the LMU:

- WWE Import Export
- Command
- SWE
- Menu

Defining the Scope of the Localization

Define what it is that you need to have localized. This could be any or all of the following:

- Software
- Strings in user interface
- Lists of values
- Seed data
- Bitmaps
- iHelp
- Training materials
- Documentation
- Online help
- Reports
- Templates and correspondence

Non-Localizable Elements

Non-localizable elements are listed below. Some of these elements are also identified in the context of export limitations for the Locale Management Utility, in "About the Localization Process" on page 79.

- DLLs
- Message files
- Log files
- Splash screens
- Help menu applets for the Siebel Web Client:
  - About Record Applet
  - About Siebel Applet
  - About SRF Applet
  - About View Applet

- Change Records Popup Applet (SWE)
- Parametric Search Form Applet
- Technical Support Applet

Developing a Glossary for Translating Product Terminology

You should develop a localization glossary. Developing a glossary of your translatable terminology offers the following advantages:

- It maintains a consistent translation of terms over the lifetime of your localization.
- It shortens the time that it takes for successive translators to do their work, because the bulk of the terminology has already been translated and they can then spend their time deliberating on how to translate terms that are new in the current release.

  NOTE: The glossary should be updated with each release to reflect new translated terms.
- You can identify terms that you do not want translated, for example, legal names of marketable products or other terms.

Table 10 on page 82 provides a sample of a localization glossary.

Table 10. Siebel Life Sciences Sample Localization Glossary

Siebel ENU Term	Siebel DEU Term	Customer-Specific Term
Account Affiliation	Firmenniederlassung	Beziehungen - Firma
Affiliations	Zugehörigkeiten	Beziehungen
Best/Visit Times	Best Times:günstigste Zeit	Besuchs- und Öffnungszeiten
Brick Subtype	Sektor Subtyp	Brick-Subtyp
Contact Affiliation	Kontaktaufnahme mit Zweigniederlassungen	Beziehungen - Person
Precall	Voranmeldung	Besuchsvorbereitung

Begin your localization effort by having your marketing department work with the translators to create this glossary. The marketing department can help the translators determine which terms should not be translated, while the translators can offer the marketing department suggestions on terminology in the target language from a localization standpoint.

About Working with Translators

This section discusses issues in choosing and training translators, and managing localization schedules in coordination with your translators.

Choosing Translators

You should consider the following questions when choosing translators to work with:

Will you use in-house translators or translation vendors?

The former may seem cheaper in the short-term, but since bilingual workers often have other responsibilities and perform translations in addition to their main functions, the results may not be as good or as sustained as hiring an outside vendor.

Another factor to consider is that just because an employee is from the country you are localizing for, it does not necessarily mean that they have the knowledge to translate your terminology correctly into the current professional vocabulary for your industry in the target locale.

If you do choose an insider for this job, make sure that he or she reads the trade publications in your field published in their native country at minimum.

There are a host of translation vendors to choose from, many of whom specialize in software internationalization, localization, or globalization, depending on your needs. (For a selection of specialists, refer to the Localization Industry Standards Association at http://www.lisa.org.)

How do you choose the right vendor?

While cost is one factor to consider, to judge whether the cost quoted is good value or not depends on the services you think you need and on what other vendors may offer for a similar cost. Find out from each vendor what their charges entail and compare each vendor's offering based on the value of the service to you as well as on cost.

For example, some vendors bundle the cost of project management into a quote, while others break out this cost. Some will charge extra for defect fixes, while some will not.

Do you require that your translators work on site or can they work remotely?

Most translation vendors use skilled specialists expert in translation for particular industries, who may be located in other U.S. states or even other countries, as opposed to in-house translators. Be prepared, whenever possible, to ship your translators flat files containing the localizable text strings and to clearly identify which strings should be translated.

Sometimes it may be most effective to work closely with qualified translators at your site. This may be appropriate when the product is being localized very rapidly. Having translators on site allows defect-fixing to occur quickly. It may also be appropriate when the product requires engineers to input localized text, as when there is no separate localization utility. Having translators on site to work together with the engineers will be highly advantageous, but is generally more expensive.

Training Translators

Even the most seasoned translators need training from you to learn about the products they are going to be localizing. Such training can include:

- Preparing a translator information kit that includes marketing brochures, MRDs, design documents, statements of direction, white papers, or other information that explains what your company sells. This provides the translators with a context for their translation work.
- How to use the application for which they are translating text strings or documentation.
- How to report product defects during localization testing. For example, truncated, localized labels in the user interface.
- How to safely enter and email translation strings in files that your engineers send using FTP to the translators and that the translators should FTP back.

Localization Schedules

Agree to a Statement of Work or timeline for the translation project early on. Understand what is involved on the translator's part, such as time needed for translation review either by vendor reviewers or by subject matter and language experts within your field offices. This makes sure that the timeline is realistic.

Consider whether your reviewers in the field consider their review of the translation a high priority when scheduling the timeline for their participation. Make sure you have their manager's commitment for participating in the review.

If your software development project incurs delays, do not expect the translators to shorten the time you have agreed to for their efforts unless you are willing to accept lower quality. Good translations take time and no machine can take the place of a human being in this regard.

Localizing Lists of Values and Multilingual Lists of Values

Lists of values (LOVs), which are used in static picklists in the Siebel application user interface, are also localized for many multilingual deployments. LOVs may also be converted into multilingual lists of values (MLOVs).

MLOVs and multilingual picklists allow values to be selected by a user in one language and retrieved by users working in other languages. The value that is stored in the database for the record is the Language Independent Code (LIC) from the LOV record, rather than the Display Value, as is true for monolingual picklists.

For MLOVs to work correctly, Language Independent Code (LIC) and Display Value must always be consistent within the same LOV type and language, as follows:

- If you have two records with the same LIC, LOV type, and language, make sure that both records point to the same display value.
- If you have two records with the same display value, LOV type, and language, make sure that both records point to the same LIC.

NOTE: For MLOVs, do not set Type to LOV_TYPE, as is done with monolingual LOVs.

As noted in "About Parameters for Language and Locale" on page 39, the Language Code parameter controls the language for MLOV display. If a Preferred Language is specified for a user, this setting overrides the parameter value to determine the MLOV language.

Some localization efforts may use choose to display LOVs/MLOVs in a common language rather than to use the same language as the application user interface. For example, for users working with a user interface in Czech (a language shipped by Siebel Systems) or Polish (an unshipped language), LOV/MLOV values may display in German, depending on the business needs of your deployment.

NOTE: If you want LOV/MLOV values to display in an unshipped language, you can copy values from one of the languages shipped by Siebel Systems and translate them into your unshipped language. In general, it is recommended to copy values from a language other than ENU. This is because many LOV values are provided for ENU that are not translated into any other language. For more information about localizing an unshipped language, see "Localizing an Unshipped Language" on page 85.

LOVs and MLOVs can also be organization-enabled, and some LOVs can also be hierarchical. If you require hierarchical MLOVs, or hierarchical organization-enabled LOVs, you must engage Siebel Expert Services both for the initial configuration and for the upgrade of these items at the next Siebel version upgrade.

For detailed information about creating and administering LOVs and MLOVs, and about converting LOVs to MLOVs, see *Configuring Siebel Business Applications* and *Applications Administration Guide*. Review all applicable guidelines and requirements.

Localizing an Unshipped Language

Topics in this section describe how to localize an unshipped language.

See also "About the Localization Process" on page 79 and "Localizing Lists of Values and Multilingual Lists of Values" on page 84.

NOTE: The process of localizing an unshipped language is considered to be customer configuration. Therefore, such configuration changes are not upgraded as part of the upgrade process described in the *Upgrade Guide*. Further configuration after an upgrade may be required in order to include new functionality for the localized unshipped language. See also Technical Note 447 on Siebel SupportWeb.

Process of Localizing an Unshipped Language

To localize an unshipped language in your Siebel application, perform the following tasks:

1. "Creating Language and Locale Records" on page 86
2. "Creating New Language Subdirectories" on page 87
3. "Creating Application Object Manager Components" on page 90
4. "Creating Virtual Directories on the Web Server" on page 91
5. "Updating the eapps.cfg File on the SWSE" on page 91

6 “Configuring Mobile Web Clients” on page 92

7 “Testing an Unshipped Language” on page 93

8 “Completing Localization for an Unshipped Language” on page 93

Creating Language and Locale Records

Localizing an unshipped language requires that appropriate Siebel language and locale records exist in the Siebel Database. If these records do not already exist, you must create them in the Siebel application. You do this in the Administration - Data screen, in the Languages and Locale views.

This task is a step in “Process of Localizing an Unshipped Language” on page 85.

Language codes used by Siebel applications use a three-letter code, such as ENU for U.S. English, FRA for French, THA for Thai, and so on. Using language codes with only two characters does not work and is not supported. See also “About Parameters for Language and Locale” on page 39.

These codes follow conventions used by Microsoft, where the first two letters represent a two-letter language code from ISO Standard 639, and the third letter differentiates the main country where this language is used. Multiple entries can represent different countries that use variants of the same language (for example, PTG for Portuguese - Portugal, and PTB for Portuguese - Brazil).

When you create language records for an unshipped language, you should use the language code names published by Microsoft. For example, the language code defined for Polish, a language that is not shipped by Siebel Systems, is PLK.

When creating language records, observe the following guidelines:

- Language codes *must* be defined using all capital letters (such as PLK rather than plk, for Polish).
- Your language code *must* use the same first two letters as the Microsoft code for the language, in order to ensure that the correct internal libraries will be used. For example, if you want to create a language code for Austrian German, name it DEA, so it will use the same code page as German (DEU). Similarly, use ENG for British English.

Siebel language records are stored in the S_LANG table.

For more information about Microsoft codes, see Microsoft documentation.

Two parameters, Language Code and (optionally) OM - Resource Language Code, are used to specify how an application uses languages. For more information, see “About Parameters for Language and Locale” on page 39.

To create a Siebel language record for an unshipped language

1 In the Siebel application, choose Navigate > Site Map > Administration - Data > Languages.

2 If necessary, create a new record for the unshipped language.

   For example, for Polish, specify Polish as the Name value and specify PLK as the Code value.

3 Start Siebel Tools against the server database, and verify that the language record displays correctly. Also verify in the List of Values Administration screen that you can enter LOV records for the new language.

NOTE: If necessary, after you create the language record, also create a record for any new locale you require. More information about locales is provided below.

About Creating a Locale

If an existing Siebel locale does not cover the users of the unshipped language you are implementing, you must also create a locale record for the location for these users.

Locales correspond to user or machine-specific settings such as Regional Settings defined in your Microsoft Windows control panel.

For example, if you create a Polish (PLK) language record, then Polish-speaking users may use an existing locale, such as Germany, or may require a locale for Poland, in order to specify settings such as time zone, dialing code, date formatting, and so on.

When creating a locale, copying an existing locale for which some of the same settings apply will make data entry easier. After you create a new locale, create translation records to provide the name for the locale for all the languages you are using.

Siebel locale records are stored in the S_LOCALE table.

See also "Additional Information About Setting Up and Administering Locales" on page 42.

For detailed information about creating locales, see the *Applications Administration Guide*.

Creating New Language Subdirectories

When you are localizing an unshipped language, you typically create new language subdirectories to be used for the unshipped language.

This task is a step in "Process of Localizing an Unshipped Language" on page 85.

If you localize the user interface and optionally MLOV values or other seed data into Polish (PLK), you would set the Language Code parameter to PLK. However, because DLLs located in language-specific subdirectories are not provided in Polish by Siebel Systems and cannot be localized into Polish, you would set the OM - Resource Language Code parameter to DEU or ENU in order to use DLLs from subdirectories for DEU or ENU. For more information, see "About Parameters for Language and Locale" on page 39.

You create language subdirectories on the Siebel Server, on the Siebel Web Server Extension (SWSE), and on the Siebel Mobile Web Client. The directory names you use must correspond to the language code you specified in "Creating Language and Locale Records" on page 86.

- For Siebel Server, see also "Creating Application Object Manager Components" on page 90.
- For SWSE, see also "Creating Virtual Directories on the Web Server" on page 91 and "Updating the eapps.cfg File on the SWSE" on page 91.
- For Mobile Web Client, see also "Configuring Mobile Web Clients" on page 92.

Table 11 on page 88 explains the language placeholders referred to in the procedures in this task and in remaining tasks listed in "Process of Localizing an Unshipped Language" on page 85.

Table 11. Language Placeholders Used in This Guide

Placeholder	Definition
XXX	The unshipped language you are localizing—for example, PLK (Polish).
YYY	The base language of the application—for example, DEU (German) or ENU (U.S. English). In this section, you are copying files from language directories for the shipped language *YYY* to the unshipped language *XXX*. **NOTE:** Alternatively, you can copy files from some other language (shipped by Siebel Systems) to the unshipped language you are localizing, rather than copy them from the base language.

To create language subdirectories for an unshipped language on the Siebel Server

1. On the Siebel Server machine, create a language subdirectory *XXX* under *SIEBSRVR_ROOT*\objects, where *XXX* is the unshipped language you are localizing, such as PLK for Polish.
2. Copy the contents of *SIEBSRVR_ROOT*\objects*YYY* into this directory, where *YYY* is the shipped base Siebel language.

   NOTE: Because you have copied the SRF file for another language, the Siebel application user interface will display in language *YYY* unless or until you compile a new SRF for the unshipped language *XXX*.
3. Create a new language subdirectory *XXX* under *SIEBSRVR_ROOT*\bin. Copy configuration files *only* from *SIEBSRVR_ROOT*\bin*YYY* into this directory. The configuration files (CFG files such as uagent.cfg for Siebel Call Center) will be used by the Application Object Managers you will create for language *XXX*.
4. For Siebel Industry Applications (SIA), create a new language subdirectory *XXX*SIA (such as PLKSIA) under *SIEBSRVR_ROOT*\bin. Copy the contents of *YYY*SIA (such as DEUSIA or ENUSIA) into this directory.

   NOTE: If you apply a patch release to the Siebel Server after completing localization, you should recopy files from *YYY*SIA to *XXX*SIA to make sure your localized product remains up to date.
5. Create a new language subdirectory *XXX* under *SIEBSRVR_ROOT*\bin\msgtmpl. Copy the contents of *SIEBSRVR_ROOT*\bin\msgtmpl*YYY* into this directory.
6. Create a new language subdirectory *XXX* under *SIEBSRVR_ROOT*\bin\reports. Copy the contents of *SIEBSRVR_ROOT*\bin\reports*YYY* into this directory.
7. Create a new language subdirectory *XXX* under *SIEBSRVR_ROOT*\webmaster. Copy the contents of *SIEBSRVR_ROOT*\webmaster*YYY* into this directory.
8. Create a new language subdirectory *XXX* under *SIEBSRVR_ROOT*\webmaster\files. Copy the contents of *SIEBSRVR_ROOT*\webmaster\files*YYY* into this directory.

9 Create a new language subdirectory *XXX* under *SIEBSRVR_ROOT*\webmaster\help. Copy the contents of *SIEBSRVR_ROOT*\webmaster\help*YYY* into this directory.

10 Create a new language subdirectory *XXX* under *SIEBSRVR_ROOT*\webmaster\images. Copy the contents of *SIEBSRVR_ROOT*\webmaster\images*YYY* into this directory.

11 Create a new language subdirectory *XXX* under *SIEBSRVR_ROOT*\webmaster\siebel_build\scripts. Copy the contents of *SIEBSRVR_ROOT*\webmaster\siebel_build\scripts*YYY* into this directory.

12 In *SIEBSRVR_ROOT*\webmaster\siebel_build\scripts*XXX*, rename the file swemessages_*yyy*.js to be swemessages_*xxx*.js, such as swemessages_plk.js for Polish.

NOTE: If you apply a patch release to the Siebel Server after completing localization, you should recopy the file swemessages_*yyy*.js from *YYY* to *XXX* (and rename the file again, as described above) to make sure your localized product remains up to date.

To create language subdirectories for an unshipped language on the SWSE

1 On the Siebel Web Server Extension (SWSE) machine, create a language subdirectory *XXX* under *SWSE_ROOT*\public, where *XXX* is the unshipped language you are localizing, such as PLK for Polish.

2 Copy the contents of *SWSE_ROOT*\public*YYY* into this directory, where *YYY* is the shipped base Siebel language.

NOTE: Each time the Web server is restarted, files and subdirectories are automatically copied from the webmaster folder on the Siebel Server to the public folder on the SWSE. For more information, see the *Siebel Installation Guide* for the operating system you are using.

To create language subdirectories for an unshipped language on a Mobile Web Client

1 On a Mobile Web Client machine, create a language subdirectory *XXX* under *SIEBEL_CLIENT_ROOT*\objects, where *XXX* is the unshipped language you are localizing, such as PLK for Polish.

2 Copy the contents of *SIEBEL_CLIENT_ROOT*\objects*YYY* into this directory, where *YYY* is the shipped base Siebel language.

NOTE: Because you have copied the SRF file for another language, the Siebel application user interface will display data in language *YYY* unless or until you compile a new SRF for unshipped language *XXX*.

3 Create a new language subdirectory *XXX* under *SIEBEL_CLIENT_ROOT*\bin. Copy configuration files *only* from *SIEBEL_CLIENT_ROOT*\bin*YYY* into this directory. The configuration files (CFG files such as uagent.cfg for Siebel Call Center) will be used by the application for language *XXX*.

4 For Siebel Industry Applications (SIA), create a new language subdirectory *XXX*SIA (such as PLKSIA) under *SIEBEL_CLIENT_ROOT*\bin. Copy the contents of *YYY*SIA (such as DEUSIA or ENUSIA) into this directory.

NOTE: If you apply a patch release to the Siebel Server after completing localization, you should recopy files from *YYY*SIA to *XXX*SIA to make sure your localized product remains up to date.

5 Create a new language subdirectory *XXX* under *SIEBEL_CLIENT_ROOT*\public. Copy the contents of *SIEBEL_CLIENT_ROOT*\public*YYY* into this directory.

Creating Application Object Manager Components

Next, you must create and configure Application Object Manager (AOM) components you are going to use for the unshipped language.

This step applies to Siebel Web Client users. It does not apply to Siebel Mobile Web Client or Developer Web Client users.

TIP: Follow this procedure for a single component for testing purposes, before you create all other components you may require.

For detailed information about creating and configuring server components, see the *Siebel System Administration Guide*.

This task is a step in "Process of Localizing an Unshipped Language" on page 85.

To configure an Application Object Manager for an unshipped language

1 Start an employee application such as Siebel Call Center in the base language, such as DEU or ENU. Navigate to Administration - Server Configuration > Enterprise Explorer.

2 In the explorer tree, expand the enterprise, then click Component Definitions.

3 In the Component Definitions list, select an Application Object Manager component on which you will base your new component. For example, select Call Center Object Manager (ENU) then choose Copy Record from the menu.

4 Provide values like the following for the server component you created:

- Component = Call Center Object Manager (PLK)
- Alias = SCCObjMgr_plk
- Component Type = Application Object Manager (the value is copied automatically from the source record you selected in Step 3)
- Component Group = Siebel Call Center
- Description = Call Center Object Manager (PLK) for Poland (the value is copied automatically from the source record you selected in Step 3)

5 Provide parameter values for this AOM component. In particular, specify the Language Code and OM - Resource Language Code parameters. Depending on the requirements for your deployment, you may set the language parameters in different ways. Verify all other parameter settings for this component.

a With the new component selected in the Component Definitions list, expand Component Definitions in the explorer tree, expand the selected Application Object Manager element, then click Parameters.

b In the Component Parameters list, query for the Language Code parameter. Set the Value to the three-letter code for the language *XXX* you are localizing, such as PLK for Polish.

c Query for the OM - Resource Language Code parameter. Set the Value to the three-letter code for the shipped language *YYY* (such as DEU or ENU).

For more information about the language parameters, see "About Parameters for Language and Locale" on page 39.

Creating Virtual Directories on the Web Server

For each new Application Object Manager component you create for an unshipped language you are localizing, as described in "Creating Application Object Manager Components" on page 90, you must create and configure a virtual directory on the Web server, such as Microsoft IIS on Windows platforms.

This task is a step in "Process of Localizing an Unshipped Language" on page 85.

You can copy an existing virtual directory entry and adapt it for the new language (for example, changing *enu* to *plk*).

For more information about creating virtual directories, see the *Siebel Installation Guide* for the operating system you are using and see Technical Note 456 on Siebel SupportWeb.

After you add the virtual directories, you update the eapps.cfg (or eapps_sia.cfg) file on the Siebel Web Server Extension (SWSE).

Updating the eapps.cfg File on the SWSE

After you created virtual directories following steps in "Creating Virtual Directories on the Web Server" on page 91, you need to create corresponding entries in the eapps.cfg file, or eapps_sia.cfg for Siebel Industry Applications, on the SWSE.

This task is a step in "Process of Localizing an Unshipped Language" on page 85.

For example, copy the section shown as [/callcenter_enu] to be [/callcenter_plk]. Change any language-specific references from ENU or enu to PLK or plk.

For example, you must specify the correct component name in the ConnectString parameter, and specify the correct directory name in the WebPublicRootDir parameter. Include all parameters in the copied section of the file.

Copy this:

```
[/callcenter_enu]
ConnectString = siebel.TCPIP.None.None://siebsrvr78:2321/sieb78/SCCObjMgr_enu
WebPublicRootDir = D:\siebel78\SWEApp\public\enu
```

To this:

```
[/callcenter_plk]
ConnectString = siebel.TCPIP.None.None://siebsrvr78:2321/sieb78/SCCObjMgr_plk
WebPublicRootDir = D:\siebel78\SWEApp\public\plk
```

Configuring Mobile Web Clients

For a Siebel Mobile Web Client or Developer Web Client, in addition to the applicable steps described in "Creating New Language Subdirectories" on page 87, you must perform the steps described here for your unshipped language.

This task is a step in "Process of Localizing an Unshipped Language" on page 85.

NOTE: The Siebel Developer Web Client is not supported for end user deployments. For more information, see the *Siebel Installation Guide* for the operating system you are using.

TIP: You can make directory and file modifications in a single product installation. For a Mobile Web Client installation, do not extract a local database for a specific user. When you have completed all configuration tasks, use Siebel Packager to create an installer package to be distributed to end users. For more information about using Siebel Packager, see *Going Live with Siebel Business Applications*.

Modifying Application Configuration Files

For any applications you will support for Mobile Web Client or Developer Web Client users, you modify the configuration files located in the new language subdirectory *XXX* under *SIEBEL_CLIENT_ROOT*\bin, which you created following steps in "Creating New Language Subdirectories" on page 87.

For example, for Siebel Call Center for Polish (PLK), edit *SIEBEL_CLIENT_ROOT*\bin\PLK\uagent.cfg to include parameter values similar to the following:

```
[Siebel]
Language = PLK
ResourceLanguage = DEU
WebClientSiteDir = D:\PROGRA~1\SIEBEL\7.8\WEBCLI~1\public\plk

...

[SWE]
MsgTemplateDir = D:\PROGRA~1\SIEBEL\7.8\WEBCLI~1\msgtempl\PLK
```

This example, where you set the Language parameter to PLK and set the ResourceLanguage parameter to DEU, ENU, or another value, supports one localization scenario. For more information, see "About Parameters for Language and Locale" on page 39 and "Creating Application Object Manager Components" on page 90.

The Language parameter is equivalent to the server parameter Language Code. The ResourceLanguage parameter is equivalent to the server parameter OM - Resource Language Code.

In the configuration file, replace other language references, such as *enu* or *ENU*, with *plk* or *PLK*, where appropriate, to support your unshipped language.

Modifying Application Shortcuts for Mobile Web Clients

For a Siebel Mobile Web Client or Developer Web Client, create application shortcuts for your unshipped language *XXX* by copying existing shortcuts for the base language *YYY*.

For example, you might copy the shortcut for Siebel Call Center - ENU and rename it Siebel Call Center - PLK. Modify the properties of this shortcut so the shortcut target is similar to the following:

```
"D:\Program Files\Siebel\7.8\web client\BIN\siebel.exe" /c
"D:\PROGRA~1\SIEBEL\7.8\WEBCLI~1\bin\PLK\uagent.cfg"
```

Testing an Unshipped Language

Now you should test the Siebel application that uses the unshipped language.

This is an interim testing phase to verify the tasks previously performed, prior to completing your localization steps, as described in "Completing Localization for an Unshipped Language" on page 93. Retest again at appropriate points to verify all remaining localization steps.

This task is a step in "Process of Localizing an Unshipped Language" on page 85.

To test the Siebel applications for the unshipped language

1. Restart the Siebel Server and the Web server to have the new settings take effect. This step also copies static files from the Siebel Server to the Web server.
2. Start the Application Object Manager for your unshipped language *XXX* (such as PLK for Polish) and test the application.

   If you copied elements from the base language *YYY* to use for the unshipped language *XXX*, the application user interface should appear in *YYY* language. The application splash screen, which is not localizable, also appears in the base language *YYY*. (For more information about non-localizable elements, see "Defining the Scope of the Localization" on page 81.)

   NOTE: If you are using multilingual list of values (MLOV) columns, make sure they are available in the appropriate values for language *XXX*.
3. Navigate to Administration - Data > List of Values and select a column that can be used as a multilingual list of values (MLOV), for example, ACCOUNT_STATUS.

Completing Localization for an Unshipped Language

To complete localization for the unshipped language *XXX*, you must translate all appropriate content that is associated with this language.

When you have translated repository elements, you would compile the SRF file for this language and distribute this file to Siebel Servers and Siebel Mobile Web Clients, into the *XXX* subdirectory for your language under the Objects directory. You created the language-specific subdirectory in "Creating New Language Subdirectories" on page 87.

This task is a step in "Process of Localizing an Unshipped Language" on page 85.

Language-specific content you can translate or create includes items listed below. See also the list of non-localizable elements in "Defining the Scope of the Localization" on page 81.

- (In Siebel Tools) Localizable Siebel Repository strings in the Siebel Database. Add translations for symbolic string references, object locales, message categories, and so on.
- (In Siebel application) Translatable language-specific values (Display Value field) for lists of values (LOVs) and multilingual lists of values (MLOVs).

  Add all values you need for your new *XXX* language. For example, LOVs used by workflow processes must be localized into the new language.

  NOTE: If an existing LOV record for your base language or another language does not have the Translate flag checked, do not copy and translate this record. Copy only those records that are flagged as translatable.

  For detailed information about configuring LOVs and MLOVs, including the use of flags such as Translate and Multilingual, see *Configuring Siebel Business Applications*.
- (In Siebel application) Other seed data such as time zone translations, iHelp files, SmartScripts, correspondence templates, email templates, Siebel Anywhere administration data, and so on.
- Text that appears in image files or Web template files.

Test all changes before you complete the project and roll out the localized application.

For more information about tasks such as these that you perform in a Siebel Tools development environment, see *Configuring Siebel Business Applications* and *Using Siebel Tools*.

About Localizing Siebel Handheld Clients

This section provides information about localizing Siebel Handheld Clients.

Siebel Handheld Clients are localized in a set of languages that is identified in *System Requirements and Supported Platforms* on Siebel SupportWeb.

The user interface (for example, screens, labels, and messages) and supporting text files and lists of values (LOVs) are fully translated.

Some areas, such as calendar date acronyms (for example, MTWTFSS for days of the week) and error messages, cannot be localized by the customer and require assistance from Siebel Global Services. This means that customers requiring Siebel Handheld Client localization in a different language cannot fully localize their applications on their own.

See also *Siebel Bookshelf* documentation for Siebel Handheld products such as Siebel Sales Handheld, including topics about end user translation capabilities.

About Supported Microsoft Windows Code Pages

For information about supported Microsoft Windows code pages for the Siebel Handheld Client, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

About Unicode Support for Siebel Handheld Clients

The Siebel Handheld Client uses Unicode internally. Therefore, the software application supports any language, as long as the handheld device is able to support data input and display for that language. (Windows CE also is built using Unicode, but there are often memory restrictions in supporting more complex languages such as Japanese on the handheld device.) For more information about supported handheld devices and operating systems, see *System Requirements and Supported Platforms* on Siebel SupportWeb.

About Adding Handwriting Recognition Software to a Supported Handheld Device

It is not recommended or supported for customers to add third-party software to a handheld device in order to allow data input in characters not supported by default on the device. The virtual memory that is available on a handheld device is significantly smaller than what is available on a desktop or laptop computer. Adding this type of software to a handheld device can result in the Siebel application being unable to load properly on the device.

About Testing Globalized Software

This section should help you test the efforts of your software internationalization and localization teams.

System Configuration

To test the configuration, you first need to establish what your key platforms are for your database servers, Siebel Servers, your clients, and your operating systems for each language you intend to deploy within your network.

Each supported platform must be tested, keeping in mind that there may be subtle differences or additional third-party software required in some cases. For example, MUI (Multilingual User Interface technology) for Windows 2000 or Windows XP may be required.

Testing for Internationalization

You should test your globally deployed applications to verify that all internationalization-related features function correctly.

For example, to test for character integrity during saving and retrieving data, you might create test cases that use a wide range of characters from your target languages. For a particular target language or set of languages, enter such data into fields and save the record. Check that you can search for the records and find them. Select the records and view them, checking that all of the data values you input are unchanged. After entering, saving, and retrieving the data records in the target language, there should be no square boxes or question marks appearing in the data.

Some of the issues you should consider when creating test cases include:

- Test for character integrity during saving and retrieving data
- Test for user interface character integrity

- Verify interaction with native language operating systems and application response to locale settings
- Verify functionality with date format (including date-time fields) and time format
- Verify functionality with time zone support
- Verify functionality with number format
- Verify functionality with buffer sizes, check for overflow
- Test file names with international characters
- Verify accelerator keys
- Verify support for euro currency and its symbol (€)
- Verify EMU triangulation conversion
- Verify decimal symbol calculations and arithmetic operations
- Verify calendar functionality
- Verify list separator
- Verify reporting and printing operations
- Verify sorting
- Verify query/find operations
- Verify phone number formats
- Verify layout for name fields, for example support for middle initial
- Verify layout for address fields, for example support for state/province
- Verify postal code is not a required field for a country with no postal codes
- Verify measurement units, metric or imperial
- Verify interaction with third-party software
- Verify import/export operations
- Verify taxation system support
- Verify translations are not truncated or otherwise corrupt
- Verify that only translatable items are translated

Testing for Linguistic Quality

Use people in your field offices whenever possible to help you check the localization for linguistic accuracy and consistency. Test any custom terms with the field, and pick terms that will be used often. For example, if you use a term like *currency rollup*, make sure it is correctly translated for your field users.

Using Automated Test Software

Third-party software is available to automate some test functions, although it does not replace the need for human testing.

Automated test software is very useful to test a wide range of characters, because an automated test can check all possible characters in Unicode, for instance. If combined with a random test data generator, it is also more likely to vary the data values much more widely than a human tester will.

On the other hand, an automated test cannot easily detect characters that have been clipped or otherwise hidden on the user interface—this is where the human eye can detect anomalies very quickly.

Of course, it is important to use automated test software that can support Unicode data values, and error messages containing failing data values. If several languages are being tested, it will be most convenient if the test software can switch between each language automatically.

Defect Reporting

Build an efficient process for defect fixing between reviewers, translators, and development.

Make sure that your defect tracking system also supports Unicode data. It is very hard to explain a problem with data values in another language if the defect system does not allow you to put the failing characters into the title or the record of the defect report.

Index