Sterling Selling and Fulfillment Foundation

Localization Guide

Release 9.1.0.2

Sterling Selling and Fulfillment Foundation

Localization Guide

Release 9.1.0.2

Contents

Chapter 1. Internationalization in Sterling Selling and Fulfillment Foundation

What Components Can Be Localized?

Every user created within IBM Sterling Selling and Fulfillment Foundation can have an associated preference for number formatting, date layout, and language. These preferences are called locale, and are identified as a pairing of a language code and a country code. Examples include `en_US` (English, US), `fr_CA` (French, Canada), `fr_FR` (French, France). A specific locale definition includes the following information:

- Country
- Language
- Date and time format
- Time zones
- Numeric Formats
- Currency
- Unit of measure for dimensions, volume, and weight

To enable the association of data such as dates, times, and strings with locale-specific formats, Sterling Selling and Fulfillment Foundation associates a locale with each user profile. This allows each user to use a locale-specific version of the product.

The locale definition associated with any organization defined in Sterling Selling and Fulfillment Foundation is used to determine only the currency and unit of measure. Date, time, and time zone information is strictly related to each user.

Table 1. What Components Can Be Localized?

Category	**Component**	**Can be localized?**	**For more details, go to**
Framework Localization		No	
Locale Specific Formats	Date/Time	Yes	"Date and Time Formats" on page 3
	Numeric Group Separator	Yes	"Localizing Number Validation" on page 29
	Numeric Decimal Separator	Yes	"Localizing Number Validation" on page 29
	E-mail	No	
	Phone Number	No	
	Credit Card Number	No	
	Currency	Yes	"Currency" on page 4
	Unit of measure for dimensions, volume, and weight	Yes	"Units of Measure" on page 6
	Time Zones	Yes	"Time Zones" on page 3

Table 1. What Components Can Be Localized? (continued)

Category	Component	Can be localized?	For more details, go to
Literals and Data	Resource Bundle	Yes	"Resource Bundles" on page 26
	Factory setup	Yes	"Localizing Factory Setup" on page 15
	Master Data - Item related	Yes	*Business Center: Localization Guide*
UI Branding	Structure of Panel Components	No	
	Localizing Themes / Cascading Style Sheets (CSS)	Yes	"User Interface Themes" on page 23
	Localizing Icons	Yes	"Localizing Images" on page 31

Literals and Data

All user interface and exception message literals are retrieved from a set of external files or database tables.

Sterling Selling and Fulfillment Foundation retrieves images and literals from locale-specific files. In order to provide a single-installation multilingual solution, Sterling Selling and Fulfillment Foundation stores multiple instances of the literals for a screen. Each instance is identified by a specific country and language pairing.

Multibyte Character Sets

Multi-byte character sets are appropriately and thoroughly taken into consideration in the database, application server, and browser tiers of Sterling Selling and Fulfillment Foundation. To represent all the characters in a language, it is sometimes necessary to use 2 (double byte) or 3 (multi-byte) bytes for each character. The longer character representations can, however, pose space and transmission challenges during application development.

- ***Double Byte Character Set (DBCS)***: One of a number of character sets defined for representing Chinese, Japanese, or Korean text (for example, JIS X 0208-1990). These character sets are often encoded in such a way as to allow double-byte character encoding to be mixed with single-byte character encoding.
- ***Multibyte Character Set (MBCS)***: A character set encoded with a variable number of bytes for each character. Many large character sets have been defined as multi-byte character sets in order to keep strict compatibility with the standards of the ASCII subset, the ISO and IEC 2022.

The Sterling Selling and Fulfillment Foundation architecture ensures that:

- All data is stored in the database using a standard compression algorithm known as UTF-8.
- The application is coded in Java, which can handle multi-byte character sets without any special changes.
- All communication between the database and the application server is through Java Database Connectivity (JDBC), which transforms the UTF-8 database representation of data to and from the multi-byte character set.

- All communication between the application server and the client is through UTF-8, which minimizes data transmission volume.
- All clients are expected to receive and send data using the UTF-8 algorithm.

Date and Time Formats

Sterling Selling and Fulfillment Foundation can present stored dates and times in any valid date or time format. Date and time fields in Sterling Selling and Fulfillment Foundation must be entered relative to the locale in which the Sterling Selling and Fulfillment Foundation database resides. Some typical date formats are as follows:

- MM/dd/yyyy
- dd/MM/yyyy
- yyyy/MM/dd

In each date format, the month can be entered as a word instead of a numeral as long as the entire length is not more than ten characters.

Some typical time formats are as follows:

- HH:mm:ss
- HH:mm

Use the hour format 'HH' as opposed to 'hh', whenever you localize Sterling Selling and Fulfillment Foundation. 'HH' signifies the 24-hour format, which is the only format supported by Sterling Selling and Fulfillment Foundation.

Time Zones

Besides being sensitive to local time zone considerations, Sterling Selling and Fulfillment Foundation is configured to recognize worldwide time zones. For example, if an order is placed in Germany for fulfillment in the United States, but, the order details are not filled on time, the software considers differences in the two time zones in order to raise an exception at the appropriate hour in the United States.

Following are the different ways in which the IBM Sterling Application Platform handles time zones:

Note: When you define a locale with a time zone (for example, en_US_EST) and when the user is logged in for that locale, the system uses the files ending in <language>_<country> code only. For instance, if the application had time zones fr_CA_EST and fr_CA_GMT set up for a locale, the system uses the same files (for example, ycpapibundle_fr_CA.properties, en_US_ycplocalizedstrings_fr_CA.properties, earth_fr_CA.css) irrespective of the time zone of the logged in user.

- When date and time values are stored, it is converted to the locale of the database. For example, assume that a database is in New York, and a customer service representative is in London. The customer service representative enters the details of an order. When the order's date and time are stored in the database that resides in New York, the values are converted to Eastern Standard Time.
- When date and time fields are displayed in the user interface, Sterling Selling and Fulfillment Foundation performs time zone calculations based on the current locale of the user, and displays the time accordingly. For example, when

a customer service representative in London views an order that resides in a database in New York, the date and time are converted from Eastern Standard Time to Greenwich Mean Time.

When a date field does not contain a time component, the time is assumed to be 12 a.m. Such fields are not adjusted for time zones when viewed from various locales with different time zones.

- The Sterling Selling and Fulfillment Foundation APIs display the date and time the way they are stored in the database. For time-sensitive fields, the time zone difference from the Universal Time Coordinate (UTC) is appended to the date and time in the output.

Numeric Formats

Numeric formats are dependent on the country and language set up in the locale definition.

Currency

Sterling Selling and Fulfillment Foundation allows each order to be processed in the preferred currency of the customer. This currency is referred to as "transactional currency".

A currency is also associated with the locale that is associated with each enterprise. This is referred to as "enterprise currency".

Order management capabilities provide order handling in multiple currencies between buyer and seller. These capabilities include a multi-currency view of an order's value as well as currency conversion procedures and rate tables.

Data structures hold flexible charge and tax taxonomies for order and invoice entities. Sterling Selling and Fulfillment Foundation has deliberately *not* taken the approach of building complex taxation rules into the application. Integration with sophisticated tax calculation programs, such as Taxware or Vertex, complement our solution and provide you with a complete taxation system. Sterling Selling and Fulfillment Foundation optionally provides a standard integration to the Taxware product line.

Note: All charges and taxes display in the user interface in the order (transactional) currency. Users can switch to view payment information of an order in either the transactional or enterprise currency.

Currency Precision

Sterling Selling and Fulfillment Foundation contains the following currency precision features:

- Unit price and unit cost can be entered and stored with a maximum of six decimal places.
- Totals are entered and stored with a maximum of two decimal places.
- Totals are rounded off (through the traditional rounding concept) if they contain 5 digits or more.

When installing Sterling Selling and Fulfillment Foundation pack for Release 8.0, you need to create your own currency or you may encounter some errors. For more information on creating currencies, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Currency Conversion

Sterling Selling and Fulfillment Foundation contains the following currency conversion features:

- Currency conversion rates are defined between two currencies and are bound by effective dates as shown in the example provided in the following table:

  Table 2. Currency Conversion Rates

From	To	From	To	Rate
British Pound	US Dollar	1/1/2004	1/31/2004	1.51
British Pound	US Dollar	2/1/2004	2/28/2004	1.56
British Pound	US Dollar	3/1/2004	3/31/2004	1.62

  A conversion rate definition is understood to imply a 1:x relationship. Thus, if one British pound is equal to 1.51 US dollars during January 2004, two British pounds are equal to 3.02 US dollars during January 2004.
- Rate definitions are not reciprocal. A conversion from US dollars to British pounds cannot use the inverse of the British pound to US dollar exchange rate. A rate for converting US dollars to British pounds must be available. Otherwise, Sterling Selling and Fulfillment Foundation reports an error. The only exception to this restriction is that a reciprocal relation does exist between the euro and its member currencies.
- Sterling Selling and Fulfillment Foundation provides the **updateConversionRates** API that allows users to import exchange rates and maintain the exchange rates in their database. Using this API, new exchange rates can be loaded every few hours if desired. You can also set up conversions using the Applications Manager.
- Sterling Selling and Fulfillment Foundation provides a user exit at the point of performing a currency conversion. This user exit allows users to perform currency conversions outside of Sterling Selling and Fulfillment Foundation. This allows real-time application of the current exchange rate instead of using the last updated exchange rate.

Currency Conversion Scenario

The following scenario illustrates how currency is displayed for an order. Suppose that a customer enters an order in France with an English company. The customer pays in francs (the transactional currency). Sterling Selling and Fulfillment Foundation converts the amount from francs to pounds using triangulation through the euro.

Because the order is placed in francs, if a company employee in England displays the order, the order is displayed in francs.

To allow auditing or reviewing of currency conversions, Sterling Selling and Fulfillment Foundation stores the exchange rates for each order as an order attribute. When an order is entered, the software stores the exchange rate used between the transactional currency and the enterprise currency. If a conversion involves triangulation, Sterling Selling and Fulfillment Foundation stores the composite exchange rate used between the euro, the transactional currency, and the enterprise currency.

Units of Measure

A Sterling Selling and Fulfillment Foundation user can select different units of measure (UOM) from various places in the Sterling Selling and Fulfillment Foundation user interface.

For supported carriers, pack dimensions entered by a shipper can be converted to different dimensions needed by the carrier. For example, if a shipper enters pack dimensions in kilos, kilos can be converted to pounds for a carrier. To accomplish this conversion for other carriers, custom coding is needed.

What Components Cannot Be Localized?

Sterling Selling and Fulfillment Foundation does not support the localization of components that are explicitly technical in nature. These components are typically used by professionals who are working with an IBM employee to perform the installation tasks, tuning tasks, and so on. The components include documentation and the following tools:

- Configuration Deployment Tool
- Configuration Data Versioning Tool
- VT220 Mobile Terminal screens, such as messages from the client and Help screen headers. (Error messages from the server, screen names, and field names can be localized.)
- JavaDocs and Entity Relationship Diagrams (ERDs)
- Installation tools and scripts (GUI-Based, Silent Install File, and Command Line)
- Upgrade scripts
- Command line tools as well as scripts, such DBVerify, agent, database extension, EAR, and WAR.
- Log file content
- Sterling Selling and Fulfillment Foundation IBM Sterling Business Intelligence reports. (IBM Sterling Warehouse Management System Sterling Business Intelligence reports can be localized.)

You can localize your Local Documentation Library implementation, and the Sterling Selling and Fulfillment Foundation documentation set. However, IBM does not provide localization support for the Online Documentation Library and the Local Documentation Library. For more information about the Documentation Library, see the *Sterling Selling and Fulfillment Foundation: Installation Guide*.

About Using Language Packs

You can localize Sterling Selling and Fulfillment Foundation in a single base language with one or multiple language packs. The base language refers to the display language of the factory setup data in the Applications Manager.

The following list defines basic terms related to language packs:

- Base Language - Refers to the language used to display factory setup data in the Applications Manager.
- Screen Literals - Refers to the labels displayed on the screens for all Sterling Selling and Fulfillment Foundation applications. These refer to the labels defined in the resource bundles property files.

- Configuration Data on Applications Manager - Refers to the factory setup data displayed on the Applications Manager screens. A specific list of fields can be translated. For example, the Status Description can be localized.
- Configuration Data on Applications - Refers to the display of the configuration data on all Sterling Selling and Fulfillment Foundation user interfaces except Applications Manager. For example, configuration data pertaining to the carrier is displayed on the console screen.

Figure 1 represents the terms described in the previous list.

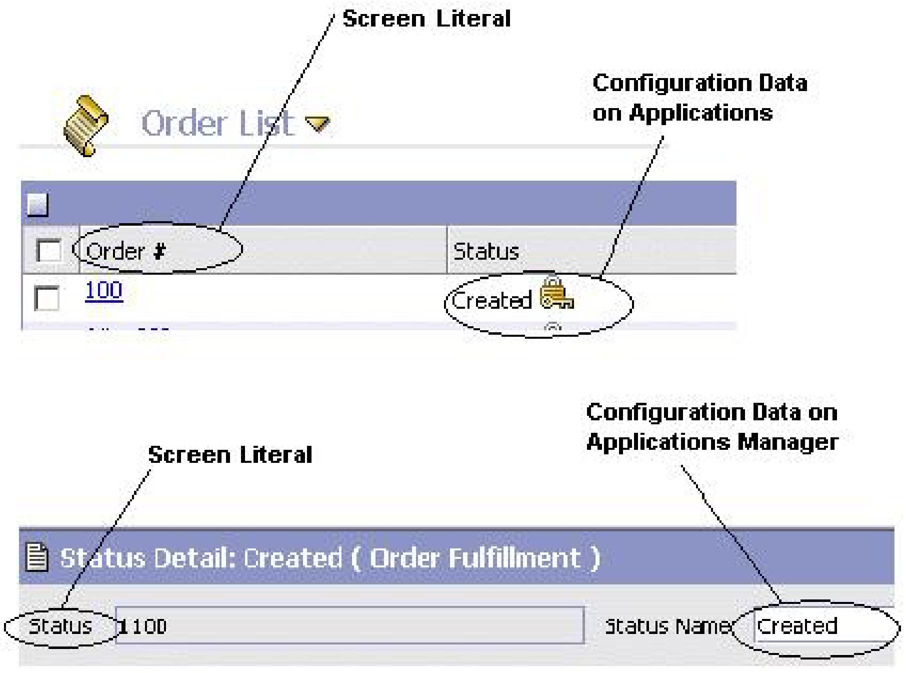
Screen Literal
Configuration Data on Applications
Order List
Order #
Status
100
Created
Configuration Data on Applications Manager
Screen Literal
Status Detail: Created (Order Fulfillment)
Status
1100
Status Name
Created

Figure 1. Schematic Diagram

Installing Language Packs in Windows XP

About this task

For language pack installations on Windows XP, where the base language is switched to a non-English language on an English operating system, the following operating system settings must be applied:

Procedure

1. Navigate to Start > Settings > Control Panel > Regional and Language Options.
2. Under Regional Options, select the applicable language.
3. Depending on the language that you are localizing, under the Languages tab, select the Install files for East Asian languages check box.
4. Click the Advanced tab and select the applicable language.

Installing Language Packs on Linux

For language pack installations on Linux, you must install the language fonts. These can be obtained from the installation disks. For example, to install Chinese fonts, install the `zh_CN-2.14-6.noarch.rpm` file.

Single-Language CD

If you get a single language pack along with a standard base language, you have two options for displaying localized information, as shown in the following table. Note that the languages listed in the table are for informational purposes only. For a complete list of available languages, contact your IBM Accounts Executive.

Table 3. Localization Options for a Single-Language CD

Options	User Locale	Screen Literals	Config Data in the Applications Manager	Config Data in the Application
Option 1: Base Language English with Japanese Language Pack	**Japanese**	Japanese	English	Japanese
	English	English	English	English
Option 2: Base Language Japanese with English Language Pack	**Japanese**	Japanese	Japanese	Japanese
	English	English	Japanese	English

Multiple-Language CDs

Each language pack is shipped as a separate CD. If you have, for example, purchased a Japanese Language Pack and Chinese Language Pack, you will have two language CDs.

Table 4 provides three localization options if you buy multiple language packs along with a standard base language.

Each of the option specified in the table is supported as part of your implementation design. However, you must decide on the best possible option based on your business requirements.

Table 4. Localization Options for Multiple-Language CD

Options	User Locale	Screen Literals	Config Data in Applications Manager	Config Data in Application
Option 1: Base Language Japanese with Chinese and English Language Pack	**Japanese**	Japanese	Japanese	Japanese
	English	English	Japanese	English
	Chinese	Chinese	Japanese	Chinese
Option 2: Base Language Chinese with Japanese and English Language Pack	**Japanese**	Japanese	Chinese	Japanese
	English	English	Chinese	English
	Chinese	Chinese	Chinese	Chinese
Option 3: Base Language English with Japanese and Chinese Language Pack	**Japanese**	Japanese	English	Japanese
	English	English	English	English
	Chinese	Chinese	English	Chinese

Chapter 2. Localizing Sterling Selling and Fulfillment Foundation

Prerequisites for Localization

Rules Governing Localization

Some of the localization rules that Sterling Selling and Fulfillment Foundation follows are governed by standards and rules external to Sterling Selling and Fulfillment Foundation, such as Java Virtual Machine (JVM), Java Runtime Environment (JRE), operating systems, and external applications. The Java programming language specifies the implementation of the locale logic. For example, see http://download.oracle.com/javase/1.5.0/docs/api/java/util/ResourceBundle.html for more information about the logic around resource bundles. For more information about specifying a JRE, see "Specifying the JRE Settings" on page 10.

Note: Use an exact match for all localization files (`_<lang>_<country>` in the file name) to ensure that all the files are correctly returned.

International Standards Organization Codes

Sterling Selling and Fulfillment Foundation uses locale-related codes as specified by the International Standards Organization (ISO). Throughout this guide, locale is represented by `<language>_<country>`. Locale comprises of `<language>`, a 2-character lower-case ISO-639 code, and `<country>`, a 2-character upper-case ISO-3166 code.

Character Encoding

When planning the localization of Sterling Selling and Fulfillment Foundation, ensure that your operating system default character encoding is set to that specified by the yfs.ui.defaultEncoding property in the `yfs.properties` file during product installation.

To modify this property, add an entry for it in the `<INSTALL_DIR>/properties/customer_overrides.properties` file. For additional information about overriding properties using the `customer_overrides.properties` file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

To enable specification of characters other than the ASCII set (for example, Asian characters and vowels with accents), ensure that the WebSphere® JVM `client.encoding.override` property is set to UTF-8, when deploying on IBM® WebSphere. For more information about the required WebSphere JVM settings, see the *Sterling Selling and Fulfillment Foundation: Installation Guide*.

Localizing Literals Displayed in External Applications

External applications (such as browsers, operating systems, application servers, or databases) used in conjunction with Sterling Selling and Fulfillment Foundation can display on-screen information. For example, Oracle WebLogic displays a status message during startup. Non-Sterling Selling and Fulfillment Foundation literals of this type can be localized. For instructions about how to localize these types of messages, see the documentation supplied by the corresponding software provider.

Microsoft Windows Clients

When localizing the Applications Manager to run on Windows, ensure that client computers have regional options set appropriately in the Windows Control panel. For example, if the client computer is running Windows and you want to display the title "Applications Manager" in Japanese, the client Windows regional options must be set to Japanese.

When localizing the Application Console to run on Windows, ensure that client computers can correctly display Unicode characters, by configuring the display from Control Panel > Display > Appearance.

Mobile Device Clients

The `ycpapibundle.properties` and `yscpapibundle.properties` files contain literals that apply specifically to mobile devices. The key for these mobile device literals are preceded with "`Mobile_`". When localizing Sterling Selling and Fulfillment Foundation to run on a mobile device, localize the literals associated with these keys.

Guidelines for Latin-1 and UTF-8 Character Encoding

When implementing Sterling Selling and Fulfillment Foundation, use the guidelines specified in this section when using Latin-1 and UTF-8 encoding.

Specifying the JRE Settings

In a multilanguage deployment, install the international version of the JRE on each client computer.

To install the plugin, download the JRE from http://www.oracle.com/technetwork/java/javase/downloads/index-jdk5-jsp-142662.html and install it on each client computer. For clients that must display non-English literals, select the "Windows (all languages, including English)" JRE.

Set Up Latin-1 Character Encoding

ISO 8859-1 is the ISO standard Latin-1 character set and encoding format. CP1252 is what Microsoft defined as the superset of ISO 8859-1. Thus, there are approximately 27 extra characters that are not included in the standard ISO 8859-1.

When using Latin-1 character encoding, it is recommended that you also refer to the following documents:

- For more information about Basic Latin and the Latin-1 Supplement that together comprise the Latin-1 (or ISO 8859-1) character set and encoding, see http://www.unicode.org/charts/.
- For an official chart download, see:

  http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=28245&ICS1=35&ICS2=40&ICS3= ().
- For a Microsoft site that lists both ISO 8859-1 and CP1252, see:

  http://www.microsoft.com/globaldev/reference/WinCP.mspx.

Set Up UTF-8 Character Encoding

About this task

Encoding must be specified for international characters to be displayed correctly on UI components. For example:

- Inventory Graph
- Delivery Map
- Shipnode in Inventory Console
- Item ID in Inventory Console

On UNIX, the UTF-8 character encoding must be specified on your UNIX application server.

To set up a UTF-8 environment:

Procedure

1. Modify all process startup scripts (for the application server, time-triggered transactions, and so on) to include the `-Dfile.encoding=UTF-8` parameter for the Java command.
2. Alternatively, when working on a UNIX application server, determine whether or not it has UTF-8 capability by running the following command:

   ```
   locale -a
   ```

   - If the command returns *any* line that indicates UTF-8, proceed to Step 1.

     For example:

     ```
     POSIX
     common
     en_US.UTF-8
     C
     iso_8859_1
     ```

   - If the command does not return any lines that indicate UTF-8, proceed to Step 3.
3. From the international language option pack appropriate to your UNIX operating system, install at least one language that has the UTF-8 character set, and then return to Step 2 to test the installation.

Database Overview

You must create the Sterling Application Platform database suitable to the data encoding format and character set used. The size of the database fields also depends on the data encoding format and type of character sets used. The following sections explain the installation and setting up of Oracle, SQL and DB2® databases.

Data Encoding Format

Sterling Selling and Fulfillment Foundation is tested and shipped using the UTF-8 transformation format. If you use a different transformation encoding format, the number of characters that you can store in standard sized database diminishes. In this case, ensure that you review and modify the Sterling Selling and Fulfillment Foundation database creation process in order to size the database fields accordingly.

Character Set

Use a character set appropriate for your localization language. For example, single-byte language character sets typically require UTF-8, while a multi-byte language may require a UTF-16 character set.

The character set you choose may impact field sizes. For example, a Varchar(40) field can only store 40/3 Japanese characters using the UTF-8 character set. This has implications on the table field sizes at the time of creation. Table creation scripts must be modified to ensure that the field lengths are correct.

Note: For the Japanese locale, the AL32UTF-8 character set or the UTF-16 character set must be used.

Running the String Length Checker

About this task

To run the string length checker for ensuring that the translated strings do not exceed the field lengths of the tables, perform the following steps:

Procedure

1. Create a folder named `/Length`.
2. Copy the contents of `<INSTALL_DIR>/repository/entity`, including all subfolders, into `/Length/entity`.
3. Copy the contents of the following subdirectories of `<INSTALL_DIR>/repository/factorysetup/` into `/Length/XMLS`:
   a. `<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS`
   b. `<INSTALL_DIR>/repository/factorysetup/sbc/XMLS`
   c. `<INSTALL_DIR>/repository/factorysetup/scecore_installation/XML`

   Note: The factorysetup directory is generated when the COPY_FCXML_TO_REPOSITORY property is set to True. For information about COPY_FCXML_TO_REPOSITORY, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.
4. Copy the `<INSTALL_DIR>/repository/datatypes/datatypes.xml` into `/Length`.
5. Copy the following JAR files to the `/Length/lib` directory:
   - `<INSTALL_DIR>`/jar/platform_afc/6_0/`platform_afc.jar`
   - `<INSTALL_DIR>`/jar/platform_baseutils.jar
   - `<INSTALL_DIR>`/jar/`install_foundation.jar`
   - `<INSTALL_DIR>`/jar/log4j/1_2_15/log4j-1.2.15.jar
   - `<INSTALL_DIR>`/jdk/jre/lib/endorsed/`xalan.jar`
   - `<INSTALL_DIR>`/jdk/jre/lib/endorsed/`xercesImpl.jar`
   - `<INSTALL_DIR>`/jdk/jre/lib/endorsed/`xml-apis.jar`
6. Copy `<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS/<language>_<country>/<baselanguage>_<basecountry>_<prefix>localizedstrings_<language>_<country>.properties` to `/Length`
7. Set `CLASSPATH= /Length/lib/xml-apis.jar;/Length/lib/xalan.jar;/Length/lib/xercesImpl.jar;/Length/lib/platform_afc.jar;/Length/lib/platform_baseutils.jar;/Length/lib/install_foundation.jar;/Length/lib/log4j-1.2.15.jar`

8. Run the following Java command. This command runs the string length checker in GENERATE mode. In this mode, the output file contains a list of translatable literals and their maximum string lengths.

   ```
   call <JAVA_HOME>/bin/java
   com.yantra.ycp.tools.localization.YCPLocalizedStringLengthTool -OUTPUT_FILE
   /Length/LengthsFile.txt -MODE GENERATE -ENTITY_DIR /Length/entity -DTYPES_FILE
   /Length/datatypes.xml -FC_DIR /Length/XMLS
   ```
9. Run the following Java command. This command runs the string length checker in CHECK mode. In this mode, the `localizedstrings` file (for instance `en_US_ycplocalizedstrings_ja_JP.properties`) is compared with the `LengthsFile.txt` file that is generated from running the string length checker in GENERATE mode. Running this tool in CHECK mode also creates `MissingLength.txt`, which contains the literals that are missing from `LengthsFile.txt`, and `MissingTranslations.txt`, which contains the literals missing from the `localizedstrings` file that was passed in the input.

   ```
   call <JAVA_HOME>/bin/java
   com.yantra.ycp.tools.localization.YCPLocalizedStringLengthTool
   -OUTPUT_FILE inconsistencies.txt -MODE CHECK -LENGTHS_FILE
   /Length/LengthsFile.txt -TRANSLATIONS_FILE /Length/
   <baselanguage>_<basecountry>_<prefix>localizedstrings
   _<language>_<country>.properties
   ```

About Unicode

Unicode is recognized as one character. For example, On\ Order\ Release\ Status\ Change is translated as Changer de statut lors du d\u00e9bloquage de la commande. The number of characters in the converted text is fifty. If you search for On\ Order\ Release\ Status\ Change, in the `LengthsFile.txt` file, the following is displayed:

```
On\ Order\ Release\ Status\ Change=50
```

This indicates that the maximum permissible length is fifty. The length of translated text exceeds the maximum permissible length by one.

About Factory Defaults

Factory defaults are limited to one language. This means that in order to switch from one language to another, you must load new factory defaults to your database.

The UI literals are not a part of the factory defaults. They can be switched from one language to another as required, as long as they have been translated appropriately.

When using an Oracle database, it is possible to use Japanese characters in English factory defaults. In fact, it could be any character in valid Unicode and UTF-8 range. When using a Microsoft SQL Server database, it is not possible to use more than one encoding and code page, based on the database collation chosen at time of the database creation. Therefore, if the database is created with a collation that is not Japanese, the factory defaults cannot contain Japanese character collation, such as Latin1 (SQL_Latin1_General_CP1_CI_AS).

Oracle Database Setup

In this release of Sterling Selling and Fulfillment Foundation, multilingual (including a multi-byte character set) environments with Oracle databases have been tested and certified using Oracle 10g created with a UTF-8 character set. See the *Sterling Selling and Fulfillment Foundation: Installation Guide* for more information about installing Oracle 10g and the settings for the UTF-8 character set.

Microsoft SQL Server Database Setup

About this task

Microsoft SQL Server has a limitation pertaining to the collation and code page with which the database is created to store characters. Because of this, during database creation, you must carefully consider collation issues pertaining to storage of data with non-English international characters, including supported Asian code pages.

It is recommended that a Microsoft SQL Server has case-insensitive collation (specified using the CI argument). For example, to select a case-insensitive collation for the Japanese language, one of the valid collation choices is Japanese_CI_AI.

The localizable factory setup is limited to the selected collation and code page because they are stored in database.

Microsoft SQL Server permits one locale for each database.

To set up Microsoft SQL Server:

Procedure

1. From the Microsoft SQL Server collation name drop-down menu, select the collation name that supports the language you want to specify. Ensure that you select a collation that is case insensitive. This creates your database.
2. Configure the `charset` value for the JDBC URL property to match the code page of the collation name selected in Microsoft SQL Server in the `<INSTALL_DIR>/properties/customer_overrides.properties` as shown below:

   The JDBC URL format is:

   jdbc:sqlserver://<DatabaseServerHostname>:<PortNumber>; DatabaseName=<Database name>?charset=<yourCharset>"
3. Install the application tables and other components, and then load the database factory defaults, as described in *Sterling Selling and Fulfillment Foundation: Installation Guide*.

DB2 Database Setup

DB2 database can be localized by passing the codeset as UTF-8. For more information about the DB2 database setup, see *Sterling Selling and Fulfillment Foundation: Installation Guide*.

Data Storage

Besides storing your transactional data, the database also stores configuration data, such as error codes and item descriptions of various attributes. This means that the database may have to store values in a language-specific format. If these database literals are not localized, screen literals are displayed inconsistently, with some of them being displayed in the localized language and the others in English.

In addition to localizing configuration data, Sterling Selling and Fulfillment Foundation enables you to localize master data, such as items, categories, and assets.

You can store item descriptions in your database in multiple languages. For more information about item descriptions, see the *Business Center: Item Administration Guide*.

Note: The localized item description is only available for Order Details and Order Line Details screens.

Localizing Factory Setup

About this task

The database factory default values can be localized for a factory setup, enabling the use of one or more locales in addition to the installed locale.

To localize the factory default XML and user-configured attributes for multiple locales:

Procedure

1. Create the necessary locale using the Applications Manager. For more information about creating a locale, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
2. Run the LocalizedStringReconciler tool from `<INSTALL_DIR>/bin` in either the extract_errorcause_db mode or the EXPORT mode for UNIX and Windows as described here. Running it in the extract_errorcause_db mode extracts the error cause and error action literals, and a `ycplocalizedstrings_xx_XX_db.properties` file is created,

   where "xx" is the language code and "XX" is the country code.

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml export
   -Ddestdir=<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml export
   -Ddestdir=<INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
   ```

   This processes the entity XMLs and identifies the missing literals for the columns that can be localized in the YFS_LOCALIZED_STRINGS table.

   The data is then exported to one or more `properties` files depending on the country, language, and variant.

   To export for a specific locale, use optional parameters language, and country. This usage is as follows:

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml export
   -Ddestdir=<INSTALL_DIR>/repository/factorysetup/complete_installation
   /XMLS -Dlanguage=fr -Dcountry=FR
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml export
   -Ddestdir=<INSTALL_DIR>\repository\factorysetup\complete_installation
   \XMLS -Dlanguage=fr -Dcountry=FR
   ```

   Note: In an instance where the YFS_LOCALE table contains two entries other than the base locale, two files are created. For example, if `fr_FR` (representing French, and France) is the locale, a file named `en_US_ycpmissinglocalizedstrings_fr_FR.properties` is created in the destination folder specified (`destdir`).

3. Edit the aa_BB_ycpmissinglocalizedstrings_xx_YY.properties file for the relevant locale, in the <INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS/<xx_YY> directory, where aa is the language code for the "From" locale, BB is the country code for the "From" locale, xx is the language code for the "To" locale, and YY is the country code for the "To" locale.

   The aa_BB_ycpmissinglocalizedstrings_xx_YY.properties file contains entries in the following format:

   ```
   Acceptance_Process=
   Add_Service=
   Line=
   ```

   Add relevant translation values to the entries. Following is an example for French literals:

   ```
   Acceptance_Process=Processus d'acceptation
   Add_Service=Ajouter service
   Line=Ligne
   ```

4. Save the modified aa_BB_ycpmissinglocalizedstrings_xx_YY.properties file in the escaped Unicode format. In this file, all the multi-byte characters should be in escaped Unicode format. The following example displays the requisite formatting for lines in this file:

   For the Japanese locale: `Add\Line=\u660e\u7d30\u306e\u8ffd\u52a0`

   For the French locale: `Invalid\inventory\operation.=Op\u00e9ration d'inventaire non valide.`

5. Run the LocalizedStringReconciler tool from <INSTALL_DIR>/bin in IMPORTTEST mode as described here. This ensures that there are no problems during the import process.

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml importtest
   -Dsrc=<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml importtest
   -Dsrc=<INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
   ```

   You can also specifically provide the Dbasefilename parameter in the localizedstringreconciler.xml file. For example:

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml importtest
   -Dsrc=<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS
   -Dbasefilename=ycpmissinglocalizedstrings
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml importtest
   -Dsrc=<INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
   -Dbasefilename=ycpmissinglocalizedstrings
   ```

   Note: The IMPORTTEST functionality of the LocalizedStringReconciler tool helps you to test the import process and roll back the changes, if required. The IMPORTTEST mode enables you to test the import process and verify if any strings cannot be added to the appropriate fields in the database for any reason, for example, extensive length.

   This is particularly important when you are upgrading from a previous release because, after a base language is switched, the import process will place the translation into the database.

The function of IMPORTTEST is similar to SWITCHTEST, in that, just as SWITCHTEST helps a new customer verify whether the switch process will be successful, IMPORTTEST helps a customer who is upgrading to verify whether the import process will be successful.

6. Run the LocalizedStringReconciler tool from `<INSTALL_DIR>/bin` in IMPORT mode as described here. This inserts the values specified in the properties file into the database.

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml import
   -Dsrc=<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml import
   -Dsrc=<INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
   ```

   You can also specifically provide the Dbasefilename parameter in the localizedstringreconciler.xml file. For example:

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml import
   -Dsrc=<INSTALL_DIR>/repository/factorysetup/complete_instal
   lation/XMLS
   -Dbasefilename=ycpmissinglocalizedstrings
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml import
   -Dsrc=<INSTALL_DIR>\repository\factorysetup\complete_instal
   lation\XMLS
   -Dbasefilename=ycpmissinglocalizedstrings
   ```

   Note: By default, when the `localizedstringreconciler` tool is run without passing the -Dbasefilename parameter, the tool picks up and runs the `ycplocalizedstrings` file, which contains the Sterling Selling and Fulfillment Foundation Factory Setup.

   Note: For customers who are upgrading from previous versions, some error codes will not be translated when a new language pack is applied, as error codes will not change and as such will not be in the latest language pack. To resolve this, you have to:

   - Run the `localizedstringreconciler` tool in export mode
   - Translate old/custom error codes
   - Run the tool in import mode to have your translations added in the database.

Extending the Default Factory-Shipped Translations

About this task

The default factory-shipped translations can be extended to create custom translations for the localization literals.

To modify the default factory-shipped translations with custom localization literals:

Procedure

1. Create a new `extn` folder in the `<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS/<language>_<country>` directory.

Note: For example, if `fr_FR` is the factory-shipped translated locale, the `extn` folder should be created in the `<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS/fr_FR` directory.

2. Copy the `<baselanguage>_<basecountry>_ycplocalizedstrings_<language>_<country>.properties` file from the `<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS/<language>_<country>` directory to the newly created `extn` folder.
3. Edit the translations in the `properties` file in the `extn` folder:
   - Modify the translations for existing localization literals with the new translations.
   - Add new localization literals and their translations, if required.
   - Remove any obsolete or unwanted translations that are not overridden for localization literals.
4. Run the LocalizedStringReconciler tool from `<INSTALL_DIR>/bin` in IMPORT mode as follows:

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml import
   -Dsrc=<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml import
   -Dsrc=<INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
   ```

   This tool first inserts the values specified in the properties file present in the `<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS/<language>_<country>` directory into the database.

   This entry is then replaced with the values specified in the properties file in the `<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS/<language>_<country>/extn` directory.

About Switching the Base Language

The base language is the language that all translations are in relation to. For example, when a new common code is added, the description is in the base language, and there are translations from this language to other languages.

Note: The base language can be switched only **once**. Switching a base language more than once may result in loss of data or other potential errors.

When this switch is performed, the following processes occur:

- Every translation to the desired base language from your current base language is changed to a translation from your desired base language to your current base language.
- Every translation from your current base language to another language is changed to a translation from your desired base language to another language.
- Every column that can be localized is converted from your current base language to your desired base language.

For example, your Sterling Selling and Fulfillment Foundation install has English as a base language, and you have set up translations to French and German. The translations are interpreted as follows:

- Translations from English to French
- Translations from English to German

By switching your base language to French, your translations are interpreted as follows:

- Translations from French to English
- Translations from French to German

  Note: In order to successfully perform a base language switch, all the localizable descriptions from the current base language to the desired base language must exist. Furthermore, there must be at least one entry in the YFS_LOCALE table corresponding to the desired base language.

Considerations When Switching the Base Language: The following points must be taken into consideration prior to performing a base language switch:

- Multiple "from" strings in a language may translate to the same "to" string in another language. During the switch, only one of these records is retained, and ambiguous records are removed.

  For example, ABC and AAC in English may both translate to FABC in French. When the base language is switched from English to French, only one record with a "from" string of FABC is retained.
- If a locale does not provide a translation, Sterling Selling and Fulfillment Foundation reverts to the base language.

  For example, there are three locales, en_US, en_GB, and fr_FR. Currently en_US is the base language. fr_FR has full translations, but en_GB only has a few translations, such as *color* to color. If the en_GB locale is used and the en_GB translation for *apple* does not exist, Sterling Selling and Fulfillment Foundation reverts to the en_US translation of *apple*.

  When the base language is switched to French, it becomes the fallback language. So, if the en_GB translation of *apple* does not exist, Sterling Selling and Fulfillment Foundation falls back on the fr_FR translation, *pomme.*

  It is recommended that *all* translations be provided for *all* locales.

  Note: When you switch the base language, the translated values may exceed the database column size, depending on the database character set (UTF-8, AL16UTF16 or any other) used. To correct the problem, adjust the translated characters to fit the database column. It is recommended that you use a character set that is appropriate to your localization requirements.

To Switch the Base Language:
About this task

To switch the base language, perform the following tasks:

Procedure

1. To ensure that there are no problems in making a base language switch, run the LocalizedStringReconciler tool from `<INSTALL_DIR>/bin` in SWITCHTEST mode as follows:

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml switchtest
   -Dlanguage=xx -Dcountry=YY
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml switchtest
   -Dlanguage=xx -Dcountry=YY
   ```

Here, xx and YY are the desired base language and country respectively, for example, fr and FR for French and France. This simulates the switch and reports any errors it finds, without making any changes to the database.

2. To perform the base language switch, run the LocalizedStringReconciler tool from `<INSTALL_DIR>/bin` in SWITCH mode as follows:

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml switch
   -Dlanguage=xx -Dcountry=YY
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml switch
   -Dlanguage=xx -Dcountry=YY
   ```

   Here, xx and YY are the desired base language and country respectively, for example, fr and FR for French and France.

   Note: Use the `<INSTALL_DIR>/properties/customer_overrides.properties` file to set the `yfs.install.localecode` property to `<your locale code>`. For more information about overriding properties using the `customer_overrides.properties` file, see the *Sterling Selling and Fulfillment Foundation: Properties Guide*.

   Note: See the *Sterling Selling and Fulfillment Foundation: Installation Guide* for information on building the search index when the base locale is switched.

Full Export to Back Up Existing Localization Literals

About this task

To create a full export to back up the existing localization literals:

Procedure

1. Run the LocalizedStringReconciler tool from `<INSTALL_DIR>/bin` in EXTRACT mode as follows:

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml extract
   -Ddestdir=<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml extract
   -Ddestdir=<INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
   ```

   This exports the literals that are currently defined in the YFS_LOCALIZED_STRINGS table.

2. To extract the existing localized strings for a specific locale, use the optional parameters `locale` and `country`. The usage is as follows:

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml extract
   -Ddestdir=<INSTALL_DIR>/repository/factorysetup/complete_installation
   /XMLS -Dlanguage=fr -Dcountry=FR
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml extract
   -Ddestdir=<INSTALL_DIR>\repository\factorysetup\complete_installation
   \XMLS -Dlanguage=fr -Dcountry=FR
   ```

   For example, if there are three locales that contain localized factory defaults in the database, one file is created for each locale in the following format:

   ```
   <baselanguage>_<basecountry>_ycpdblocalizedstrings_<language>_
   <country>_db.properties
   ```

Each file is saved within the `extn` folder within the respective locale directory. For instance, for the locale `fr_FR`, the file `en_US_ycpdblocalizedstrings_fr_FR_db.properties` is created and saved to the `<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS/fr_FR` directory.

3. To re-import the backed-up literals, run the LocalizedStringReconciler tool from `<INSTALL_DIR>/bin` in IMPORT mode with the file for which you have exported the literals as follows:

   For UNIX:

   ```
   sci_ant.sh -f localizedstringreconciler.xml import
   -Dsrc=<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS
   ```

   For Windows:

   ```
   sci_ant.cmd -f localizedstringreconciler.xml import
   -Dsrc=<INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
   ```

Deleting Unused Localization Literals

Procedure

To delete unused localization literals, run the LocalizedStringReconciler tool from `<INSTALL_DIR>/bin` in EXPORT mode as follows:
For UNIX:

```
sci_ant.sh -f localizedstringreconciler.xml export
-Ddestdir=<INSTALL_DIR>/repository/factorysetup/complete_installation/XMLS
-Ddelete=true
```

For Windows:

```
sci_ant.cmd -f localizedstringreconciler.xml export
-Ddestdir=<INSTALL_DIR>\repository\factorysetup\complete_installation\XMLS
-Ddelete=true
```

This deletes the unused literals when exporting the literals that are currently defined in the YFS_LOCALIZED_STRINGS table.

Localizing Master Data for a Multi-Language Installation

Sterling Selling and Fulfillment Foundation enables the localization of master data such as items, categories, and assets. Localizing configuration data allows administrators to only configure localized literals for words and phrases. For example, if you localize the configuration data for a word Store, the localized literal for Store is displayed in every field that uses the word Store. However, master data localization allows you to specify the localized literal, for example, Super Computer, as an item's description. In this example, the localized literal Super Computer is displayed for the specified item's description. Administrators can use IBM Sterling Business Center to localize master data in each locale and to modify master data localizations.

Data Localization in Entities for Locale-Specific Information

In addition to localized screen labels, localized values for the actual content (such as transaction data) must be stored and appropriate translation values displayed, according to the business usage or meanings across different locales.

To enable the display of localized data, the entity framework supports a child entity that stores different locale values for each column.

Columns which contain localized values must be marked as *Localized* in the parent entity. Such columns automatically become a part of the child entity.

Note: You must provide LANGUAGE, COUNTRY and VARIANT columns for the child entity. You must also define the primary key, indices and relationship between an entity and its localized description. Since the LANGUAGE, COUNTRY and VARIANT columns along with the primary key of the parent table are used to search for the localized values of a column, you must define a unique key for this combination.

Displaying Locale-Specific Information

About this task

To display locale-specific information:

Procedure

1. A field called **LocaleDescriptionForEntity** is added to entity definition, which is used to store localized description in each child table. Each child table is identified by this field.
2. Set the value of this field to the name of the parent entity. This value should match the value of *Name* attribute in the parent entity.

   For example, if the entity definition to store localized values for a column in the parent table "PLT_QUALIFIED_TAG" is:

   ```
   <Entity Cacheable="true"
      Description="Stores qualifier information"
      EntityType="CONFIGURATION" AuditRequired="N"
      Module="ycp" Name="QualifiedTag" Prefix="PLT_"
      TableName="PLT_QUALIFIED_TAG" XMLName="QualifiedTag">
   <Attributes>
   ....
   ```

   The corresponding child entity definition should be:

   ```
   <Entity Description="Stores qualifier information"
      EntityType="CONFIGURATION" Name="Qualifier_Locale"
      Prefix="YFS_" TableName="YFS_QUALIFIER_LOCALE"
      XMLName="QualifierLocale"
      LocaleDescriptionForEntity="QualifiedTag">
      ...
   ```

   You can define child tables for Platform or lower stack entities and mark columns as localized. Each entity can have only one localized child entity.
3. Another field **DisplayLocalizedFieldInLocale** is added to the API layer. Pass this field to an API to indicate the locale from which the localized values must be displayed. The value of this field must be in the format, *language_country_variant (*fr_FR, en_US_CA):

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <QualifiedTag QualifiedTagId="" QualifiedTagKey="key1"
   DisplayLocalizedFieldInLocale="fr_FR"/>
   ```
4. The **get** method for localized fields is modified to return the local values from the child tables, instead of the actual value. This method reads the locale set in the **DisplayLocalizedFieldInLocale** field and uses the language, country and variant information to obtain the corresponding values from the child table. For example, `get QualifiedTagDescription()`

   If there is no corresponding localized value, then the actual value from the parent entity is displayed.
5. To obtain the actual value (non-localized values) from the parent entity instead of the localized one, use the method `getNonLocalized`. For example, `getNonLocalized QualifiedTagDescription()`

Searching for the Localized Data

You can search for the localized value only in limited scenarios.

The screens that support the indexed searches automatically enable the search by localized value, as the index includes the localized value.

The `DisplayLocalizedFieldInLocale` input variable normally controls the output behavior. In some cases and depending on the API behavior, you can also search for the localized value. For more information about the API behavior, see the *Sterling Selling and Fulfillment Foundation: Javadocs.*

Directory Structure and File Names

The Sterling Selling and Fulfillment Foundation directory structure must be maintained based on the language and country localization. File names are limited to one locale and can only contain characters `a` through `z`, `0` (zero) through `9`, and the underscore (_).

Likewise, because many files are created using a transaction ID, this rule also applies to transaction names and transaction IDs. Similarly, you should adhere to these limitations when localizing the template file name for transaction events (`TransactionID.On_SUCCESS.xml`).

Sterling Selling and Fulfillment Foundation follows the hierarchy of lookups provided here:

1. When a certain locale is selected in the console, it searches for some files with that particular suffix specified in the locale. For example, if the locale `ja_JP` is chosen in the console, the system searches for files such as `ycpapibundle_ja_JP.properties`, `validation_ja_JP.properties`, `sapphire_ja_JP.css`, and so forth.
2. If these files are not found, the systems looks at the default locale of the operating system that the application server is running. If it is en_US, the system tries to look for the files mentioned in Step 1, but with the suffix en_US, for example, `ycpapibundle_en_US.properties`.
3. Only if the first two steps fail are the default files, for example, `ycpapibundle.properties` used.

The method of defaulting to the standard files occurs for the files that can have locale suffix such as `alertmessages.js`, `scfoundationiconsbe.jar`, and so on.

User Interface Themes

The user interface theme files specify the screen colors and display fonts to be used. Display fonts are dependent on the languages that must be supported. Some fonts may not support all the languages. For example, Tahoma (the Sterling Selling and Fulfillment Foundation default font) does not support Japanese; for best results when localizing Sterling Selling and Fulfillment Foundation to Japanese, use the Microsoft Gothic font. When setting up a theme, choose a font that displays the specific language you require. For example, when setting up a Japanese locale, customize the theme to use a font that displays Japanese characters such as Hiragana.

Note: If you use a font that is bigger than the Sterling Selling and Fulfillment Foundation default font (Tahoma), it may be necessary to customize the `<INSTALL_DIR>/repository/datatypes/datatypes.xml` file to increase the user interface size of data types that are used for input fields in the Application Console. In particular, the user interface size of the "Date" data type should be increased.

When choosing a font for a specific language, you can refer to Table 5 for the language's recommended font.

Table 5. Recommended Fonts for Languages

Language	Font
French	Tahoma
German	Tahoma
Japanese	MS UI Gothic
Spanish	Tahoma
Simplified Chinese	SimSun
Traditional Chinese	PMingLiu
Korean	Dotum

Localizing Themes

About this task

When localizing your Applications Manager user interface themes, you modify the theme-specific XML file. When localizing your Application Console user interface themes, you modify the theme-specific CSS file. For example, the following files must be localized for themes:

```
<INSTALL_DIR>/repository/xapi/template/merged/resource/<theme>.xml and
<INSTALL_DIR>/repository/eardata/platform/war/css/<theme>.css
```

These files are localized by appending the language and country codes in the file name. For example, if you are using the sapphire theme in a French locale, localize the following files:

```
<INSTALL_DIR>/repository/xapi/template/merged/resource/sapphire_fr_FR.xml
<INSTALL_DIR>/repository/eardata/platform/war/css/sapphire_fr_FR.css
<INSTALL_DIR>/repository/eardata/smcfs/war/css/sapphire_ssdcs_fr_FR.css
<INSTALL_DIR>/repository/eardata/smcfs/war/css/sapphire_ssdcs_override_fr_FR.css
```

The following themes are distributed with Sterling Selling and Fulfillment Foundation:

- Earth
  - `<INSTALL_DIR>/repository/xapi/template/merged/resource/earth.xml`
  - `<INSTALL_DIR>/repository/eardata/platform/war/css/earth.css`
  - `<INSTALL_DIR>/repository/eardata/smcfs/war/css/earth_ssdcs.css`
  - `<INSTALL_DIR>/repository/eardata/smcfs/war/css/earth_ssdcs_override.css`
- Jade
  - `<INSTALL_DIR>/repository/xapi/template/merged/resource/jade.xml`
  - `<INSTALL_DIR>/repository/eardata/platform/war/css/jade.css`
  - `<INSTALL_DIR>/repository/eardata/smcfs/war/css/jade_ssdcs.css`
  - `<INSTALL_DIR>/repository/eardata/smcfs/war/css/jade_ssdcs_override.css`
- Sapphire
  - `<INSTALL_DIR>/repository/xapi/template/merged/resource/sapphire.xml`

- `<INSTALL_DIR>/repository/eardata/platform/war/css/sapphire.css`
- `<INSTALL_DIR>/repository/eardata/smcfs/war/css/sapphire_ssdcs.css`
- `<INSTALL_DIR>/repository/eardata/smcfs/war/css/sapphire_ssdcs_override.css`

Note: For locale specific theme files, the CSS entry for detailpagetitle6 for button panel must be adjusted to fit the translation.

To localize a theme:

Procedure

1. Copy the `<INSTALL_DIR>/repository/xapi/template/merged/resource/<theme>.xml` file and save it as `<INSTALL_DIR>/repository/xapi/template/merged/resource/<theme>_<language>_<country>.xml`.
2. Copy the `<INSTALL_DIR>/repository/eardata/platform/war/css/<theme>.css` file and save it as `<INSTALL_DIR>/repository/eardata/platform/war/css/<theme>_<language>_<country>.css`.

   Edit the `<INSTALL_DIR>/repository/eardata/platform/war/css/<theme>_<language>_<country>.css` file to change the display font for the Application Console. In addition, the font name and size for the graph displayed in the Inventory Summary screen in the Inventory Console is configured in the `<INSTALL_DIR>/repository/xapi/template/merged/resource/<theme>_<language>_<country>.xml` file.

   For example, in the default `sapphire.xml` file, the graph font is configured as:

   ```
   <!-- Font for Inventory Graphs(Axis Titles & Labels) -->
   <Font Name="InvGraphFont" FontName="Tahoma" FontSize="12"/>
   <!-- Font for Inventory Graphs -->
   ```

   To localize double-byte languages such as Japanese, IBM recommends that you edit the `<INSTALL_DIR>/repository/xapi/template/merged/resource/<theme>_<language>_<country>.xml` file to use either the MS UI Gothic or SimSun font as follows:

   ```
   <!-- Font for Inventory Graphs(Axis Titles & Labels) -->
   <Font Name="InvGraphFont" FontName="simsun" FontSize="12"/>
   <!-- Font for Inventory Graphs -->
   ```
3. Rebuild the `resources.jar` by running the following command from the `<INSTALL_DIR>/bin` directory:

   ```
   /deployer.sh -t resourcejar
   ```
4. If you are using Oracle WebLogic or IBM WebSphere, rebuild the EAR.

   Note: If your application server is running on UNIX, the valid fonts that you can use are stored in the `<JAVA_HOME>/jre/lib/font_properties.<file.encoding>` file.

Literals

All Sterling Selling and Fulfillment Foundation components use a common resource bundle that contains literals displayed on the screens. Sterling Selling and Fulfillment Foundation enables you to customize and localize resource bundles as required.

In addition, literals used in customized screens have their own resource bundle and should also be considered during the localization process. For more information about localizing your Sterling Selling and Fulfillment Foundation customizations, see the *Sterling Selling and Fulfillment Foundation: Customizing Console JSP Interface for End-User* .

Note: Literals cannot be localized in:
- Condition Builder
- Order/Shipment Monitor
- Hard-coded literals in APIs

For a complete list of resource bundle literals, along with the screens on which those literals appear, see the `<DocumentationCD>/resource_mapping.htm` file.

Resource Bundles

Complete resource bundles are released in the `ycpapibundle.properties` and `yscpapibundle.properties` files with the localized versions of Sterling Selling and Fulfillment Foundation. Incremental updates are not provided. If you localize Sterling Selling and Fulfillment Foundation, it is your responsibility (or that of your third-party localization company) to compare and validate the differences between the resource bundles shipped with the product to those you have localized.

The resource bundles of Sterling Selling and Fulfillment Foundation are located in the `<INSTALL_DIR>/resources/ycpapibundle.properties` and `<INSTALL_DIR>/resources/yscpapibundle.properties` files.

Localizing Resource Bundles

About this task

To localize the resource bundles:

Procedure

1. Copy the `<INSTALL_DIR>/resources/ycpapibundle.properties` file and save it as `<INSTALL_DIR>/resources/ycpapibundle_<language>_<country>.properties`.
2. Copy the `<INSTALL_DIR>/resources/yscpapibundle.properties` file and save it as `<INSTALL_DIR>/resources/yscpapibundle_<language>_<country>.properties`.
3. Each resource bundle contains a `<key>=<value>` pair, where `key` is the resource key and `value` is the literal displayed for the corresponding locale. Replace `<value>` with the translated value.
   - When localizing menus in the Applications Manager, by default, the accelerator key is the first character in a menu item. To specify any other character as the accelerator key, insert an ampersand (`&`) just before that character.
   - When localizing the console UI, be aware that changing the height or width of the text in the application may affect the layout of the screens. It may be necessary to customize certain screens to achieve optimal layout after the other localization steps are complete. For example, if the resource bundle contains translated literals that are lengthy, you may have to increase the width of the screen in order to accommodate the larger size of the translated literal.
   - Some of the literals that have to be translated in the resource bundles contain data place holders. These data place holders indicate that the literal is displayed with one or more data values within the literal. For example, when the application displays the error message "Priority should be greater than `X`",

where X could be any number. Because the location of "X" within the literal can be different for different languages, the resource bundle uses a place holder that can be placed anywhere in the literal during translation. The resource bundle entry looks like this:

```
PRIORITY_ERROR_MESSAGE=Priority should be greater than {0}
```

Notice how the "{0}" place holder indicates where the dynamic data value appears in the literal. This "{0}" can be placed anywhere in the literal, for example, the following two options are valid possibilities:

```
PRIORITY_ERROR_MESSAGE=A number greater than {0} should be entered
```

or

```
PRIORITY_ERROR_MESSAGE={0}: Priority entered should be greater than this
```

Placeholders give you the flexibility to translate the literal in any way the corresponding language dictates. Note that multiple place holders may appear in the literal as well, for example, {0}, {1}, {2}, and so on. Each place holder must exist somewhere in the corresponding translated literal.

- When using literals that contain data place holders, you cannot use single quotation marks. If a single quotation mark is used in conjunction with a place holder, the single quotation mark is not displayed and the place holder is not replaced with its replacement value. In order to avoid this situation, enter two single quotes wherever a single quote is required. For example, the following statement is invalid:

  ```
  PRIORITY_ERROR_MESSAGE=The primary organization's name is {0}
  ```

  However, the following statement is valid:

  ```
  PRIORITY_ERROR_MESSAGE=The primary organization''s name is {0}
  ```

- Files should be returned in native format with UTF-8 encoding.
- Properties should be returned in escaped Unicode format with UTF-8 encoding.

4. The default font used is Tahoma. Therefore if you want to display or type Unicode characters, you should localize the theme. This is done by changing the font to Unicode in the theme-specific XML files.
5. Save the modified file. If the file is in UTF-8 format, convert it to ASCII by running the native2ascii command as follows:

   ```
   native2ascii -encoding UTF-8 <source file> <target file>
   ```

   The file should be returned in the following format:

   ```
   <filename>_<2 letter code for language as given by ISO 639>_<2 letter code
   for territory as given by ISO 3166>.<file extension>
   ```

   For example, ycpapibundle.properties should be returned as ycpapibundle_fr_FR.properties and yscpapibundle.properties should be returned as yscpapibundle_fr_FR.properties.
6. If you are customizing Sterling Selling and Fulfillment Foundation, save the extended resource bundles as <INSTALL_DIR>/resources/extn/extnbundle_<language>_<country>.properties.

   For example, ycpapibundle.properties should be saved as ycpapibundle_fr_FR.properties and yscpapibundle.properties should be saved as yscpapibundle_fr_FR.properties.
7. For extended tag attributes, add the following bundle entry in extnbundle.properties for each extended tag attribute:

   ```
   Item_Tag_<TagName>=<Tag Name>
   ```

8. Create the resource jar using the ./deployer.sh -t resourcejar.
9. If you are using Oracle WebLogic or IBM WebSphere, rebuild the EAR.

Note: The Custom_Code_Prefix and Custom_Code_Postfix properties in the `ycpapibundles.properties` file are used to prefix or append identifying literals or extensions to your newly created custom transaction IDs, supply types, demand types, or document types. When you create a new transaction ID, supply type, demand type, or document type, the value specified for these properties is prefixed or appended to each of these types of literals when displayed in the user interface. The default value for the Custom_Code_Prefix is "" (blank) and the Custom_Code_Postfix is ".ex". You can change this value if it does not suit your requirements.

Validating Date, Time, and Number

Date and number validations are performed using JavaScript. By default, Sterling Selling and Fulfillment Foundation provides validation for the `en_US` locale (English for the United States).

Localizing Date and Time Validation

About this task

Dates can be stored in a standard format, but displayed according to the required format. If the date is entered on a screen, it must be verified. The date format is specified in the Applications Manager Locale Details screen. For more information, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

To localize date and time validation:

Save the `<INSTALL_DIR>/repository/eardata/platform/war/yfcscripts/Validation.js` file as `Validation_<language>_<country>.js` (with all translations in escaped Unicode format) and make modifications to the new file as indicated in the following steps.

Procedure

1. Ensure that the date and time values match the entries specified in the Locale fields in the Applications Manager. For detailed information about the Locale fields and their suggested syntax, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide* .
2. Change the [`yfcDateFormat`] (MM/dd/yyyy), [`yfcTimeFormat`] (HH:mm:ss) and [`yfcDateTimeFormat`] (MM/dd/yyyy HH:mm:ss) variables to contain the correct date and time format.

   These date and time formats are according to the United States English version. When you create a `Validation.js` file for another language, these formats change accordingly. The following table specifies the date and time formats.

Table 6. Date and Time Variable Formats

Date Element	Description
yyyy	Four-digit year, for example, 2009
MM	Two-digit month, for example, 05 to indicate May
dd	Two-digit day, for example, 01 to indicate the first day of the month
mm	Minutes
HH	Hours
ss	Seconds

3. Localize the pop-up messages and calendar formats as described in “Localizing Calendar Pop-Up Windows in the Console” on page 30.

Localizing Number Validation

About this task

By default, Java displays the localized number format based on what is specified for `<language>_<country>` in the Applications Manager. For the Sterling Selling and Fulfillment Foundation UI to validate numeric values, the application must read the values as specified in the `Validation_<language>_<country>.js` file. This means that the validation JavaScript file must contain validation logic that is specific to the number format to be used for the validation for each locale.

To localize number validation:

Procedure

1. Save the `<INSTALL_DIR>/repository/eardata/platform/war/yfcscripts/Validation.js` file as `Validation_<language>_<country>.js` (with all translations in escaped Unicode format).
2. Change the decimal separator and grouping separator, as required. The grouping separator indicates how to format numbers visually; it has no impact on the actual value of the number.

   For example, `en_US` uses a comma (",") and `fr_FR` uses a non-breaking space (" "). When specifying the separator, use the Unicode literal.

   For example, the non-breaking character for fr_FR will be specified as shown below:

   ```
    yfcGroupingSeparator = "\u00a0";
   ```
3. Localize the exception messages for invalid number format as described in “Literals” on page 25.
4. Run the resource deployer from `./deployer.sh -l info -t resourcejar.`
5. If you are using Oracle WebLogic or IBM WebSphere, rebuild the EAR.

Templates

E-Mail Templates

You can store e-mail templates using any character encoding format, but the encoding format must be set by configuring the yfs.email.template.encoding property in the `<INSTALL_DIR>/properties/customer_overrides.properties` file.

For additional information about overriding properties using the `customer_overrides.properties` file, see *Sterling Selling and Fulfillment Foundation: Properties Guide*.

Exception Alert Templates

Exception alert templates enable you to provide additional text to the alerts raised. This enables you to make error messages more descriptive and easy to understand. They also provide a means of providing a hyperlink to the resolution screens from the Alert Console.

For example, for any alert created for an order, shipment, or load document type, a hyperlink is created and displayed in the "Created For" column in the Alert List screen. You localize the literals displayed in this column by translating them in the

`DefaultListTemplate.xsl` file located in your `<INSTALL_DIR>/repository/xapi/template/merged/exception_console` directory.

You can store exception alert templates using any character encoding format, but the encoding format must be configured in the yfs.file.encoding property in the `<INSTALL_DIR>/properties/customer_overrides.properties` file. If these properties are not explicitly set in the `<INSTALL_DIR>/properties/customer_overrides.properties` file, Sterling Selling and Fulfillment Foundation uses UTF-8 character encoding.

Displaying a Translated Default Organization Name

About this task

In the Applications Manager, some screens display the organization code instead of the organization name in the title. Because organization codes are not translated, these are displayed in English. To have translated organization names displayed instead of organization codes, override the default for the yfs.showOrganizationName.enabled property in yfs.properties. You can do this by adding an entry to the customer_overrides.properties file.

Procedure

To override the property, add the following line to customer_overrides.properties: `yfs.yfs.showOrganizationName.enabled=Y`.
See the *Sterling Selling and Fulfillment Foundation: Properties Guide* for more information about properties files.

Localizing Calendar Pop-Up Windows in the Console

About this task

In the Console, calendar pop-up windows contain literals that must be localized.

Only the literals in the calendar pop-up window can be localized. The order of the days cannot be localized. Therefore, the week always starts with Sunday.

To localize date formats for calendar popup windows in the Console:

Procedure

1. Copy the `<INSTALL_DIR>/repository/eardata/platform/war/common/alertmessages.js` file and save it as `<INSTALL_DIR>/repository/eardata/platform/war/common/alertmessages_<language>_<country>.js` (with all translations in escaped Unicode format).
2. Copy the `<INSTALL_DIR>/repository/eardatasmcfs/war/ysc/scripts/yscalertmessages.js` file and save it as `<INSTALL_DIR>/repository/eardatasmcfs/war/ysc/scripts/yscalertmessages_<language>_<country>.js` (with all translations in escaped Unicode format).
3. Copy the `<INSTALL_DIR>/repository/eardata/smcfs/war/ysc/scripts/yscalertmessages.js` file and save it as `<INSTALL_DIR>/repository/eardata/smcfs/war/ysc/scripts/yscalertmessages_<language>_<country>.js` (with all translations in escaped Unicode format).
4. Translate the following date-related literals:
   - `monthArray` - Contains the series of literals to be used for months of the year

- `weekdayList` - Contains the series of literals to be used when displaying the literals for days of the week
- `weekdayArray` - Contains the series of literals to be used when displaying the shortened literals for days of the week
- `todayString` - Displays the word "Today"

5. If you are using Oracle WebLogic or IBM WebSphere, rebuild the EAR.

Localizing Message Pop-Up Windows in the Console

About this task

In the Console, message pop-up windows contain literals that must be localized.

To localize message pop-up windows in the Console:

Procedure

1. Copy the `<INSTALL_DIR>/repository/eardata/platform/war/common/alertmessages.js` file and save it as `<INSTALL_DIR>/repository/eardata/platform/war/common/alertmessages_<language>_<country>.js`, (with all translations in escaped Unicode format).
2. Copy the `<INSTALL_DIR>/repository/eardata/smcfswar/ysc/scripts/yscalertmessages.js` file and save it as `<INSTALL_DIR>/repository/eardata/smcfswar/ysc/scripts/yscalertmessages_<language>_<country>.js` (with all translations in escaped Unicode format).
3. Copy the `<INSTALL_DIR>/repository/eardata/smcfs/war/ysc/scripts/yscalertmessages.js` file and save it as `<INSTALL_DIR>/repository/eardata/smcfs/war/ysc/scripts/yscalertmessages_<language>_<country>.js` (in UTF-8 encoding format).
4. Edit your exception messages in the files, as required.

Localizing Images

Images and icons are stored in JAR files. A separate JAR file can be used for each locale. Both the Applications Manager and Application Console use their own mechanisms for reading images.

Localize Application Consoles Images

Procedure

To localize images used in the Application Console, copy the `<INSTALL_DIR>/repository/eardata/config/war/yfscommon/smcfsiconsbe.jar` file to `<INSTALL_DIR>/repository/eardata/config/war/yfscommon/smcfsiconsbe_<language>_<country>.jar` file and make the necessary locale-specific changes to the images.

Results

The warning message that is displayed when you click the Sterling Selling and Fulfillment Foundation icon in the top right-hand corner of the Application Console user interface, is included as text in an image file (`about_box_back.gif`). You can localize this message by following the instructions in this section.

Localize Applications Manager Images

About this task

To localize the images used in the Applications Manager, you can either:

- Copy the `<INSTALL_DIR>/repository/eardata/config/war/yfscommon/smcfsicons.jar` file to `<INSTALL_DIR>/repository/eardata/config/war/yfscommon/smcfsicons_<language>_<country>.jar` file and make the necessary locale-specific changes to the images.

  Important: You must also remove the RSA encryption files (STERLING.SF and STERLING.RSA) located in the META-INF directory.
- Copy the images in `<INSTALL_DIR>/repository/eardata/config/war/yfscommon/smcfsicons.jar` file to a temporary directory and make the necessary locale-specific changes to the images. Rename the directory to `smcfsicons_<language>_<country>.jar` and copy it to `<INSTALL_DIR>/repository/eardata/config/war/yfscommon`.

Results

The warning message that displays at the bottom of the Applications Manager **About box** when you select **Help > About** from the Applications Manager menu is included as text in an image file (`about.gif`). You can localize this message by following the instructions in this section.

Localizing the Greex File

About this task

A Greex file, also known as advanced XML file, contains the advanced XML condition or Greex Rule that is defined by a user. By localizing the Greex file, you can localize an advanced XML condition or Greex Rule.

To localize a Greex file:

Procedure

1. Create the BundleResolver class and implement the following methods within the class:
   - getString(String key) method and return the localized strings.

   For example, to localize a Greex file using properties files:
   ```
   public class MyBundleResolver implements BundleResolver
   {
      Properties prop = new Properties();
        public MyBundleResolver()
        {
          //read and initialize the property file
        }
        public  String getString(String key)
        {
            Return prop.getProperty(key);
        }
    }
   ```
2. Register the BundleResolver class with the GreexContext using the registerBundle() method, for example:

```
public class MyApp
{
  GreexContext  ctx = new GreexContext();
  ctx.registerBundle(new MyBundleResolver())
}
```

Localizing the Context-Sensitive Help

About this task

During installation, the Context-Sensitive Help is placed under the `<INSTALL_DIR>/xapidocs/online_help` directory.

To localize the Context-Sensitive Help of the Sterling Application Platform:

Procedure

1. Create a directory for the desired locale. For example, if you are translating the Context-Sensitive Help to German, create the following directory for files created in the `online_help` directory:

   `<INSTALL_DIR>/xapidocs/online_help/de_DE/`
2. Copy the files in the `<INSTALL_DIR>/xapidocs/online_help/` to the locale-specific directory.
3. Translate the files in the locale-specific directory.
4. Add `yfs.onlinehelp.path.overrideforlocale.<locale>` to the `customer_overrides.properties` file.

   For translating to the German locale, set the `yfs.onlinehelp.path.overrideforlocale.de_DE` property to `/smcfsdocs/yfscommon/online_help/de_DE`.
5. Re-build and deploy the Sterling Selling and Fulfillment Foundation EAR. For more information on how to build an EAR, see the *Sterling Selling and Fulfillment Foundation: Properties Guide* .

Results

If you are installing both Sterling Selling and Fulfillment Foundation and the Language Pack together, a one-time creation and deployment of the EAR file is sufficient. If you have already deployed your application and are planning to install the Language Pack later, you must re-create and redeploy the EAR file.

For more information about creating and deploying the EAR file for your chosen application server, see the *Sterling Selling and Fulfillment Foundation: Installation Guide*.

Note: If you do not want to translate the Context-Sensitive Help, you can use the default US English files, but the non-translated files must be copied to the locale-specific directory, as described in Step 2. If this is not done, a user who is logged in under a non-US locale will receive an "Error 404--Not Found" message when accessing the Context-Sensitive Help.

Localizing the Synonyms for Catalog Search

About this task

Sterling Selling and Fulfillment Foundation allows you to configure synonyms for catalog search. Synonyms enable a catalog search to return an expanded list of items based on related search terms. For example, if notebook is configured as a synonym for laptop, and laptop is entered as a search term, the search results return items identified as notebooks as well as laptops.

By default, Sterling Application Platform does not provide synonyms for search terms. However, you can configure synonyms for each of the locales your catalog search supports. For example, if your catalog search supports the US-English and French locales, you can configure US-English and French synonyms.

To define synonyms for a locale:

Procedure

1. In the extended XML configuration file for item search, specify a path for the synonym file in the Locale element. For information about customizing item search, see the *Extending the Database Guide*.
2. In the `Properties` directory, open the synonym file that corresponds to the file specified in the Locale element of the extended XML configuration file for item search.
3. Specify related terms using the following format:
   term1=synonym1, synonym2, synonym3
   term2=synonym4, synonym5
   For example, to configure coffee maker as a synonym for coffee machine, specify:
   coffee machine=coffee maker

   Note: The mapping of related terms for synonyms is not reciprocal. In the example provided previously, coffee maker is a synonym for coffee machine, but coffee machine is not specified as a synonym for coffee maker. If a customer enters coffee machine as a search term, the search returns results pertaining to both coffee machine and coffee maker. However, if a customer enters only coffee maker as the search criteria, the search returns only coffee maker.

Chapter 3. Localizing the Rich Client Platform Application

Prerequisite Guidelines from the International Standards Organization

Rich Client Platform uses locale-related codes as specified by the International Standards Organization (ISO). Throughout this guide, locale is represented by `<language>_<country>`. Locale comprises of `<language>`, a 2-character lower-case ISO-639 code, and `<country>`, a 2-character upper-case ISO-3166 code.

Note: For complete steps to localize the Rich Client Platform Application, see the Implementation Guide specific to your PCA.

Localizing User Interface Themes for Rich Client Platform Applications

About this task

The user interface theme files specify screen colors, display fonts, and images to use. Display fonts are dependent on the languages that are supported. Some fonts may not support all languages. When setting up a theme, select a font that displays the specific language you require or select an image that is displayed for a particular locale. For example, when setting up a Japanese locale, customize the theme to use a font that displays Japanese characters, such as Hiragana.

When localizing your Rich Client Platform user interface themes, localize the Rich Client Platform-specific theme files and the Rich Client Platform application-specific theme files.

The Rich Client Platform contains `com.yantra.yfc.rcp.common_<theme_name>.ythm` theme file, which is located in the `com.yantra.yfc.rcp.common_<version>` directory that is available in the `<INSTALL_DIR>/repository/rcp/rcpclient/com.yantra.yfc.rcp_<version>.zip` file, once extracted.

The Rich Client Platform contains `<Plug-in_ id>_<theme_name>.ythm` theme file, which is located in the `plugins/com.yantra.pca.ycd.rcp_<version>` directory within the `<INSTALL_DIR>/repository/rcp/rcpclient/com20.zip` file.

The theme file is localized by appending the language and country codes in the file name. For example, if you are localizing the sapphire theme file in a French locale of the Rich Client Platform, modify the `com.yantra.yfc.rcp.common_sapphire.ythm` file as:

```
com.yantra.yfc.rcp.common_sapphire_fr_FR.ythm
```

Procedure

To localize the sapphire theme file in a French locale of the Rich Client Platform application, modify the `<Plug-in_id>_sapphire.ythm` file as:

```
<Plug-in_id>_sapphire_fr_FR.ythm
```

Results

For example, if you are localizing the sapphire theme file in a French locale of the IBM Sterling Call Center and IBM Sterling Store PCA, modify the `com.yantra.pca.ycd.rcp_sapphire.ythm` file as:

```
com.yantra.pca.ycd.rcp_sapphire_fr_FR.ythm
```

Localizing Themes for Fonts

Theme entries for the Font theme are specified in the Font, ForegroundColor, and BackgroundColor elements under the ThemeEntry element. There are three types of theme entries for the Font theme:

- Font—The Font element contains data pertaining to the font's height, name, and style. Table 7 describes the attributes of the Font element.

Table 7. Font Element Attribute List

Attribute	Description
Height	Enter the height of the font.
Name	Enter the font name, for example, Tahoma, Arial, and so on.
Style	Enter the font style, for example, NORMAL, BOLD, and so on.

- ForegroundColor and BackgroundColor—The ForegroundColor element describes the color of the text that is displayed. The BackgroundColor element describes the background on which the text is displayed. Available attributes for both elements are red, blue, and green. These standard RGB values must be entered in the decimal color code (0-255).

Localizing Themes for Images

About this task

The theme entries for the Image theme are specified in the Image element under the ThemeEntry element. In the Image element, specify the path for the locale-specific image to be used in the theme in the `Path` attribute.

To store the locale-specific images:

Procedure

1. Create an `icons` folder in the `<INSTALL_DIR>/repository/rcp/extn` directory.
2. Put all the locale specific images in this newly created `<INSTALL_DIR>/repository/rcp/extn/icons` folder.

Results

Following is a sample data from the `<Plug-in_ id>_<theme_name>.ythm` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Theme id="sapphire">
  <ThemeEntry Name="Label">
     <Font Height="9" Name="Tahoma" Style="NORMAL"/>
     <ForegroundColor Blue="0" Green="0" Red="0"/>
     <BackgroundColor Blue="255" Green="255" Red="255"/>
  </ThemeEntry>
  <ThemeEntry Name="CComboEditor">
     <Font Height="8" Name="Tahoma" Style="NORMAL"/>
     <BackgroundColor Blue="255" Green="255" Red="255"/>
     <ForegroundColor Blue="0" Green="0" Red="0"/>
  </ThemeEntry>
  <ThemeEntry Name="Text">
     <Font Height="8" Name="Tahoma" Style="NORMAL"/>
     <ForegroundColor Blue="0" Green="0" Red="0"/>
     <BackgroundColor Blue="255" Green="255" Red="255"/>
  </ThemeEntry>
 <ThemeEntry Name="Composite">
```

```
        <Font Height="10" Name="Tahoma" Style="NORMAL"/>
        <BackgroundColor Blue="255" Green="255" Red="255"/>
        <ForegroundColor Blue="198" Green="146" Red="140"/>
    </ThemeEntry>
    <ThemeEntry Name="DateLookup">
        <Image Path="/icons/calendar.gif"/>
    </ThemeEntry>
    <ThemeEntry Name="HeaderTriangle">
        <Image Path="/icons/header_triangle.jpg"/>
    </ThemeEntry>
```

Localizing Themes on the Client Application

About this task

To localize the client application theme files, perform the following tasks:

Procedure

1. Navigate to the `<INSTALL_DIR>/repository/rcp/rcpclient/` directory.
2. Extract files from the `com.yantra.yfc.rcp_<version>.zip` file to the `<extracted_files>` directory of your choice.

   All Rich Client Platform-specific theme files will now be located in the `<extracted_files>/com.yantra.yfc.rcp.common_<version>` directory.

   The theme files are in the following format:

   `com.yantra.yfc.rcp.common_<theme_name>.ythm`

   Here, `<theme_name>` refers to the name of a particular theme. By default, Rich Client Platform provides three different theme files named sapphire, jade, and earth.
3. Copy the `com.yantra.yfc.rcp.common_<theme_name>.ythm` file to the `resources` directory under the `extensions` folder that you created. For more information about creating an extensions folder, see the *Sterling Selling and Fulfillment Foundation: Installation Guide*.
4. Navigate to the `<INSTALL_DIR>/repository/rcp/rcpclient/` directory.
5. Extract the `<PCA_APPLICATION_ID_VERSION>.zip` file to the `<extracted_files>` directory of your choice. Navigate to the following directory:

   `<extracted_files>/plugins/<Plug-in_id>_<version>`

   In the `<Plug-in_id>_<version>` directory, there is a `<PCA_APPLICATION_ID_VERSION>.jar` file. The theme file is in the root directory of this JAR file and is called `<Plug-in_id>_<theme_name>.ythm`.
6. Copy the `<Plug-in_id>_<theme_name>.ythm` file to the `resources` directory under the extensions folder that you created.

   For example, if you are localizing a theme file of the Sterling Call Center and Sterling Store PCA, extract the `<INSTALL_DIR>/repository/rcp/rcpclient/com20.zip` file.
7. Navigate to the `<extracted_files>/plugins/com.yantra.pca.ycd.rcp_<version>` directory.

   Here, <extracted_files> refers to the directory in which you have extracted the com20.zip file

   In the `com.yantra.pca.ycd.rcp_<version>` directory, there is a `com20.jar` file. The theme file is in the root directory of this JAR file and is called `com.yantra.pca.ycd_<theme>.ythm.`
8. Copy the `com.yantra.pca.ycd_<theme>.ythm` file to the `resources` directory under the `extensions` folder that you created.

9. Rename the `<Plug-in_ id>_<theme_name>.ythm` files as `<Plug-in_ id>_<theme_name>_<locale_name>.ythm` and the `com.yantra.yfc.rcp.common_<theme_name>.ythm` files as `com.yantra.yfc.rcp.common_<theme_name>_<locale_name>.ythm`.

   The `<locale_name>=lang_cc`, where `lang` refers to the language code and `cc` refers to the country code. For example, `com.yantra.pca.ycd_ sapphire_en_US.ythm`.
10. Modify the theme entries in the theme file based on the user's locale. For more information about theme entries in the theme file, see “Localizing Themes for Fonts” on page 36 and “Localizing Themes for Images” on page 36.
11. Create a new icons folder and store all the images or icons that you want to localize in this folder. Now, copy the icons folder to the `<RCP_EXTN_FOLDER>/ resources` directory.
12. If you are using Oracle WebLogic or IBM WebSphere, rebuild the EAR.

Results

Note: For information on localizing theme files for PCAs, see the Implementation Guide specific to your PCA.

Using Resource Bundles to Localize Literals in RCP Applications

All Rich Client Platform applications use a resource bundle that contains literals or text displayed on the screens. Rich Client Platform enables you to customize and localize resource bundles based on a user's locale.

In addition, literals used in customized screens have their own resource bundle and should also be considered during the localization process.

All Rich Client Platform application plug-ins contain bundle files named as `<Plug-in_id>_<name>.properties`. The bundle file contains `<key>-<value>` pairs, and specifies resources such as control text, string, and so on that have to be localized in the `<Key>` and the localized string or text in `<value>`.

Following is an example of `<key>=<value>`:

```
my_name=Rich Client Platform Application
```

The resource bundles of a Rich Client Platform application are located in its archive file.

For example, the resource bundle for the Sterling Call Center and Sterling Store PCA is located in the `<INSTALL_DIR>/repository/rcp/rcpclient/com20.zip` file.

Extract files from the `com20.zip` file to the `<extracted_files>` directory of your choice. Navigate to the following directory:

```
<extracted_files>/plugins/com.yantra.pca.ycd.rcp_<version>
```

Within this directory, there is a `com20.jar` file. The bundle files are in the root directory of this JAR file and are called `com.yantra.pca.ycd_bundle.properties` and `com.yantra.pca.ycd_Messages_bundle.properties`.

Localizing the Resource Bundles in RCP Applications

About this task

To localize the resource bundles:

Procedure

1. Copy the `<Plug-in_id>_<name>.properties` file from the Rich Client Platform application plug-in's directory of the application that you want to localize, and also from the Rich Client Platform plug-in directory to the `resources` directory under the `extensions` folder that you created.

   Note: Because every Rich Client Platform application is dependent on the Rich Client Platform plug-in, whenever you want to localize a Rich Client Platform application, you must modify the `<Plug-in_id>_<name>.properties` file of both the Rich Client Platform plug-in and the Rich Client Platform application plug-in directories. The Rich Client Platform plug-in directory is located in the `<INSTALL_DIR>/repository/rcp/rcpclient/com.yantra.yfc.rcp_<version>.zip` file.

   To localize the Sterling Call Center and Sterling Store PCA bundle file, follow the instructions provided in the example below.
2. For example, if you are localizing the bundle file of the Sterling Call Center and Sterling Store PCA, the `<INSTALL_DIR>/rcp/COM/<version_no>/COM/rcpclient/<OS>/com.zip` file (where `<version_no>` is the release number) will contain the Sterling Call Center and Sterling Store PCA plug-in directory called `com.yantra.pca.ycd.rcp_<version>` directory.
3. Extract the `com.zip` file to the `<extracted_files>` directory of your choice. Navigate to the following directory:

   `<extracted_files>/plugins/com.yantra.pca.ycd.rcp_1.0.0`

   Within this directory, there is a `com20.jar` file. The bundle file is in the root directory of this JAR file and is called `com.yantra.pca.ycd_<name>.properties`.

   Therefore, copy the `<Plug-in_id>_<name>.properties` file from both the Sterling Call Center and Sterling Store PCA plug-in directory as well as from the Rich Client Platform plug-in directory to the `resources` directory under the `extensions` folder that you created. For more information about creating the `extensions` folder, refer to the *Sterling Selling and Fulfillment Foundation: Installation Guide*.
4. Rename the `<Plug-in_id>_<name>.properties` file as `<Plug-in_id>_<name>_<locale_name>.properties`. The `<locale_name>=lang_cc`, where `lang` refers to language code and `cc` refers to the country code, for example, `com.yantra.pca.ycd_bundle_en_US.properties`.
5. Each resource bundle contains a `<key>=<value>` pair where `key` is the resource key and `value` is the literal displayed for the corresponding locale. Replace the `<value>` with the translated value.

   Note: By default, Rich Client Platform localizes:
   - Text on Labels
   - Table Column names
   - Descriptions in Combo Boxes
   - Text on Buttons
   - Tab Folder items
   - Groups names

Rich Client Platform does not localize the text in the text boxes and the keys used for identification, such as ItemId or a resource key.

Following are the sample bundle entries from the `*.properties` file:

```
Credit_Card_#=Credit Card
View_Details=View &Details
Customer_Address=Customer Address
Save=&Save
Ship_To_Address=Ship To Address
Address=Address
Close=&Close
```

Here, entries on the left represent the resource key and entries on the right represent the translated value that is displayed for each control, text, or string, based on the user's locale.

6. If you want to get the localized value for any key, use the following method:

   ```
   YRCPlatformUI.getString(String bundleKey);
   ```

   It returns the localized string as the output.

Results

Note: For information on localizing resource bundles for PCAs, see the Implementation Guide specific to your PCA.

Localizing the Resource Bundles in Eclipse

About this task

To localize the Eclipse platform resource bundles:

Procedure

1. Modify the Rich Client Platform application's `*.ini` file to provide the appropriate program arguments to use the language pack. You can find the `*.ini` file for the Rich Client Platform application in the `<INSTALL_DIR>/bin/rcpdrop/<OPERATING_SYSTEM>/<PCA_DIR>/` directory.

   Here, `<INSTALL_DIR>` refers to the directory in which you have installed Sterling Selling and Fulfillment Foundation.

   For example, if you want to run the Sterling COM PCA in debug mode, edit the `<INSTALL_DIR>/bin/rcpdrop/<OPERATING_ SYSTEM>/com/com.ini` file.
2. In the `*.ini` file, add one of the following arguments:
   - Program arguments

     ```
     -nl
     <locale_code>
     ```
   - VM arguments

     ```
     -Duser.language=<language_code>
     ```

   Note: You must enter the program arguments before the VM arguments. Also, ensure that the program arguments are separated by a line break.

Chapter 4. Localizing XML Attributes in Factory Setups

Localizing XML Attributes in Factory Setups

About this task

Table 8 on page 42 enables you to determine the XML file name associated with the corresponding XML attributes. This table lists the factory default XML attributes that can be localized along with the associated database table name. The full name of the XML file can be derived from the database table name using the procedure described here.

To derive a list of XML files:

Procedure

1. Find the database table that corresponds to the XML attribute you want to translate.
2. Using the database table name, append "`.xml`" to it and prepend the appropriate prefix for the module to which it belongs. Module prefixes are as follows:
   - BI—Sterling Business Intelligence
   - INV—IBM Sterling Global Inventory Visibility
   - OMD—Order Management, IBM Sterling Supply Collaboration, and IBM Sterling Reverse Logistics.
   - OMP—Order Management Platform
   - OMR—Order Management Returns
   - OMS—Order Management Services
   - RDT—Rapid Deployment
   - REF—Reference implementation
   - VAS—Value Added Services
   - WMS—Warehouse Management Services
   - YCM—Catalog Management
   - YCP/PLT—Platform
   - YCS—Carrier Service
   - YDM—Delivery Management
   - YSC—Supply Chain

Results

For example, the `Actionname` attribute corresponds with the YFS_ACTION database table, which uses the `YCP_YFS_ACTION.xml` file.

Database Tables and XML Attributes That Can Be Localized

Table 8 enables you to determine the XML file name associated with the corresponding XML attributes. This table lists the factory default XML attributes that can be localized along with the associated database table name. The full name of the XML file can be derived from the database table name using the procedure described in “Localizing XML Attributes in Factory Setups” on page 41.

Table 8. Localizable Factory Setup XML Attributes

Database Table Name	**XML Attribute Name**
SI_VERSION	ProductLabel
YFS_ACTION	Actionname GroupId
YFS_ACTIVITY	Description
YFS_ACTIVITY_CONSTRAINT	ConstraintDescription
YFS_ADAPTER	AdapterDescription
YFS_ADJUSTMENT_REASON	Description
YFS_ALLOCATION_RULE	Description
YFS_ANSWER_OPTION	Description
YFS_ATP_RULES	AtpRuleName
YFS_ATTR_ALLOWED_VALUE	ShortDescription LongDescription
YFS_ATTRIBUTE	ShortDescription LongDescription
YFS_ATTRIBUTE_DOMAIN	ShortDescription LongDescription
YFS_ATTRIBUTE_GROUP	ShortDescription LongDescription
YFS_BARCODE_TRANSLATION	Description
YFS_BASE_ACTIVITY_GROUP	ActivityGroupName
YFS_BASE_DOCUMENT_PARAMS	Description
YFS_BASE_EVENT	EventName
YFS_BASE_PACK_DO_NOT_MIX	FieldNameDesc
YFS_BASE_PROCESS_TYPE	ProcessTypeName
YFS_BASE_PURGE_CRITERIA	PurgeCodeDescription
YFS_BASE_SHIP_CONSTRAINTS	FieldNameDesc
YFS_BASE_TRANSACTION	BaseTranname
YFS_BUSINESS_DOCUMENT	DocumentDescription DocumentName
YFS_CALENDAR	CalendarDescription
YFS_CARRIER_SERVICE	CarrierServiceDesc

Table 8. Localizable Factory Setup XML Attributes (continued)

Database Table Name	XML Attribute Name
YFS_CATEGORY	ShortDescription Description
YFS_CATEGORY_DOMAIN	ShortDescription Description
YFS_CHARGE_CATEGORY	Description
YFS_CHARGE_NAME	Description
YFS_CLASSIFICATION_PURPOSE	ClassificationPurposeDesc
YFS_COMMON_CODE	CodeLongDescription CodeShortDescription **Note:** For entries where CODE_TYPE starts with CODE_TYPE_LIST, change the CODE_DESCRIPTION property in the `ycpapibundle.properties` and `yscpapibundle.properties` files.
YFS_CONDITION	ConditionName
YFS_CONFIG_VERSION_LABEL	Description
YFS_COUNT_PROGRAM	CountProgramName
YFS_COUNT_PROGRAM_COND	Description
YFS_COUNT_STRATEGY	Description
YFS_CURRENCY	CurrencyDescription
YFS_DATE_TYPE	Description
YFS_DEPARTMENT	DepartmentName
YFS_DEVICE_SUB_TYPE	Description
YFS_DEVICE_TYPE	Description
YFS_DOCUMENT_FORMAT	FormatDescription
YFS_DOCUMENT_PARAMS	Description
YFS_ENTERPRISE	Enterprisename
YFS_EQUIPMENT_TYPE	Description
YFS_ERROR_CAUSE_ACTION	Cause Action
YFS_ERROR_CODE	ErrorMessage
YFS_EVENT	EventName ExceptionType
YFS_FLOW	FlowName FlowGroupName
YFS_EXCEPTION_TYPE	ExceptionTypeDescription
YFS_EXECUTION_EXCEPTION	ExceptionShortDescription ExceptionLongDescription

Table 8. Localizable Factory Setup XML Attributes (continued)

Database Table Name	XML Attribute Name
YFS_FREIGHT_TERMS	Description ShortDescription
YFS_HM_GROUPS	GroupName
YFS_HOLD_TYPE	HoldTypeDescription
YFS_IBA_SEQUENCE_HDR	Description
YFS_INVENTORY_DEMAND_TYPE	Description
YFS_INVENTORY_STATUS	Description
YFS_INVENTORY_SUPPLY_TYPE	Description
YFS_ITEM_ATTR	ItemAttributeDescription
YFS_ITEM_UOM_MASTER	Description
YFS_LINE_RELATIONSHIP_TYPE	RelationshipShortDescription RelationshipLongDescription
YFS_LOCALE	LocaleDescription
YFS_LOCATION_SIZE	Description
YFS_MONITOR_TYPE	Field1Name, Field2Name, Field3Name, Field4Name, Field5Name, Field6Name, Field7Name, Field8Name, Field9Name, Field10Name
YFS_MONITORING_CONSOLIDATION	Description
YFS_NEGOTIATION_RULE	NegotiationRuleId
YFS_NODE_TYPE	NodeTypeDescription
YFS_ORDER_LINE_TYPE	LineTypeDesc
YFS_ORDER_TAG_DETMN	Description
YFS_ORGANIZATION	OrganizationName **Note:** OrganizationCode and OrganizationKey cannot be localized.
YFS_PAYMENT_TYPE	PaymentTypeDescription
YFS_PICK_STRATEGY	Description
YFS_PIPELINE	PipelineDescription
YFS_PLA_ACTIVITY_DETER	Description
YFS_PLA_ZONE_SET	Description
YFS_PRINT_DOCUMENT	PrintDocumentDescription
YFS_PROCESS_TASK_TYPE	ProcessTaskTypeDesc
YFS_PROCESS_TYPE	ProcessTypeName Description
YFS_PRODUCTIVITY_TYPE	Description ProductivityType
YFS_PURGE_CRITERIA	PurgeCodeDescription ErrFileName LogFileName

Table 8. Localizable Factory Setup XML Attributes (continued)

Database Table Name	**XML Attribute Name**
YFS_QUESTION	QuestionText
YFS_QUEUE	QueueDescription **Note:** QueueKey and QueueId cannot be localized.
YFS_REGION	RegionDescription
YFS_REGION_LEVEL	AddressFieldAlias
YFS_REGION_MATCH_PREF	AddressFieldAlias
YFS_RETRIEVAL_STRATEGY	Description
YFS_RETURN_DISPOSITION	Description
YFS_ROLE	RoleDescription **Note:** RoleID and RoleKey cannot be localized.
YFS_ROLE_DOCUMENT	DocumentName DocumentDescription
YFS_RULES	RuleSetFieldDescription
YFS_SCAC	ScacDesc **Note:** Scac and ScacKey cannot be localized.
YFS_SCAC_AND_SERVICE	ScacAndService ScacAndServiceDesc
YFS_SERVICE_SLOT	ServiceSlotDesc
YFS_SERVICE_SLOT_GROUP	ServiceSlotGroupDesc
YFS_SERVICE_TYPE	Description
YFS_SHIP_NODE	Description
YFS_SHIPMENT_GROUP	Description
YFS_SPECIAL_SERVICES	SpecialServicesDescription
YFS_STATUS	Description StatusName
YFS_STATUS_MODIFICATION_TYPE	ModificationLevelScreenName ModificationTypeScreenName
YFS_TASK_TYPE	TaskTypeName
YFS_TRANSACTION	Tranname
YFS_TRANSACTION_DEPENDENCY	TransactionDependencyName
YFS_TRANSACTION_DEPENDENCY_GRP	GroupID GroupDescription
YFS_UOM	UomDescription **Note:** If `YFS_UOM.xml` is changed, the `YFS_UOM_CONVERSION.xml` should also have corresponding changes.
YFS_USER	Username
YFS_USER_EXIT_IMPL	ImplementationNotes
YFS_USER_GROUP	UsergroupName
YFS_VOICE_APPLICATION	Description

Table 8. Localizable Factory Setup XML Attributes (continued)

Database Table Name	**XML Attribute Name**
YFS_VOICE_APPLICATION_INSTANCE	Description
YFS_VOICE_WORK_FLOW	Description
YFS_WAVE_SIZE_CONSTRAINT	Description
YFS_ZONE	Description
YPM_PRICELIST_HDR	Description
YPM_PRICING_RULE	Description

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

1623-14, Shimotsuruma, Yamato-shi

Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be

Trademarks

Index