Configuration Guide for Siebel Offline Client for Life Sciences

Version 8.0

February 2011

ORACLE

Contents

Chapter 4: Installing and Setting Up Oracle JDeveloper and Adobe Flash Builder

Chapter 5: Getting Started with Oracle JDeveloper

Chapter 6: Managing String Translations

Chapter 7: Adding and Using Icons

Chapter 8: Extending the Basic Content in the Sales Application

Chapter 9: Extending the Business Functions

Chapter 10: Configuring Top-Level Objects

Appendix A: Configuration Files, Filter Definitions, and Calculated Expressions

Index

1 What's New in This Release

What's New in Configuration Guide for Siebel Offline Client for Life Sciences, Version 8.0

This new guide describes how to install and use the configuration projects and the utilities plug-in for Oracle JDeveloper 11*g* supplied with Siebel Offline Client for Life Sciences to configure, customize, and extend the component applications (such as Sales, Sync, and Asset Message Planner).

NOTE: The Offline Client for Life Sciences Release 1.5 supports Siebel CRM Version 8.0.

Siebel Offline Client for Life Sciences, which includes mobile Customer Relationship Management (CRM) capabilities, supports sales representatives in the field by allowing them to plan and execute sales calls, drop product samples, obtain associated signatures, and deliver electronic presentations.

Table 1 lists the chapters in this version of the documentation to support Version 8.0 of the software.

Table 1. New Product Features in Configuration Guide for Siebel Offline Client for Life Sciences, Version 8.0

Topic	Description
Chapter 2, "Overview of the Offline Client for Life Sciences,"	This chapter provides an overview of the user interface and architecture of the Offline Client for Life Sciences.
Chapter 3, "Installing the Required Software for the Configuration Environment"	This chapter describes how to install and set up the required software for the configuration environment for the Offline Client for Life Sciences.
Chapter 4, "Installing and Setting Up Oracle JDeveloper and Adobe Flash Builder"	This chapter describes how to install and configure the Offline Client for Life Sciences.
Chapter 5, "Getting Started with Oracle JDeveloper"	This chapter describes how to get started with some basic configuration tasks.
Chapter 6, "Managing String Translations"	This chapter describes how to manage string labels, including how to translate the strings into a language other than English.
Chapter 7, "Adding and Using Icons"	This chapter describes how to remove, replace, or add additional icons to the user interface (UI).
Chapter 8, "Extending the Basic Content in the Sales Application"	This chapter describes how to extend the basic content by displaying or hiding the data that is provided with the Offline Client for Life Sciences.
Chapter 9, "Extending the Business Functions"	This chapter describes how to extend the Offline Client for Life Sciences by adding custom functions that meet specific business requirements.

Table 1. New Product Features in Configuration Guide for Siebel Offline Client for Life Sciences, Version 8.0

Topic	Description
Chapter 10, "Configuring Top-Level Objects"	This chapter describes how to extend the Offline Client for Life Sciences by making available additional object types to meet specific business requirements.
Appendix A, "Configuration Files, Filter Definitions, and Calculated Expressions"	This appendix contains reference information that is required to perform the tasks in this guide.

2 Overview of the Offline Client for Life Sciences

This chapter provides an overview of the Offline Client for Life Sciences, its user interface (UI), its architecture, and its basic operations. It contains the following topics:

NOTE: Before you begin using this book to configure and customize the Offline Client for Life Sciences, it is recommended that you understand the preconfigured features of the Offline Client for Life Sciences. For more information, see *Getting Started with Siebel Offline Client for Life Sciences*.

About the Offline Client for Life Sciences

The Offline Client for Life Sciences is a mobile application that is designed to use metadata that supports application configuration. Customers can configure it to meet their specific requirements. The configuration projects and the utilities plug-in for Oracle JDeveloper 11*g* provide an application development environment from which you can configure the component applications for the Offline Client for Life Sciences.

NOTE: You can only use the Oracle JDeveloper utilities plug-in with the Offline Client for Life Sciences Release 1.5; the previous Utilities application for Release 1.0 will not work with the Offline Client for Life Sciences Release 1.5.

The following components make up the Offline Client for Life Sciences:

- **Client applications.** The client applications include the Sales application and the Asset Message Planner application:
  - The Sales application allows sales representatives to manage sales visits and deliver electronic presentations to customers through its user interface that is designed and optimized for mobile PC pen- and touch- based computing.
  - The Asset Message Planner application allows marketing and sales users to manage assets and message plans. These client applications operate offline and use the local database, supporting database queries and Create, Read, Update and Delete (CRUD) operations. They also provide local client business services, such as validation rules and scripting logic.

- **Offline database.** The client applications use offline databases (also called local databases) to store the relevant data from the Siebel CRM Server, which allows users to work uninterrupted. If a user makes a change to a local database, the change is uploaded to the application server when network connectivity resumes. No user intervention is required during data synchronization between the local databases and the application server.

  NOTE: User intervention is required only if a conflict between the local database and the Siebel CRM database occurs.

- **Sync application.** The Sync application creates the local database, manages the overall synchronization process, and sends and retrieves data from Siebel CRM. The synchronization process is automated so that the end user rarely sees or manages the process directly. The Sync application detects network connectivity and performs synchronization at predefined intervals. It manages the connectivity, authentication, and data transfer to Siebel CRM. It also creates local databases, composes, and decomposes Web services, and tracks changes made to the local database.

- The following application:

  Siebel CRM. The source for the data downloaded to and the destination for changes made locally to the Offline Client for Life Sciences. Oracle's Siebel CRM Web services interface allows the downloading and uploading of data. For more information about Siebel CRM Web services, see *Siebel CRM Web Services Reference* on Siebel Bookshelf.

  NOTE: The Siebel Bookshelf is available on Oracle Technology Network (OTN) and Oracle E-Delivery. It might also be installed locally on your intranet or on a network location.

- **Utilities plug-in and a set of configuration projects.** The Offline Client for Life Sciences is designed to use metadata that supports application configuration. A set of configuration projects and a utility plug-in for Oracle JDeveloper allow you to configure your own version of the Offline Client for Life Sciences for deployment. The configuration projects are different depending on whether you intend to connect to Siebel CRM.

  NOTE: Siebel CRM does not publish all Web services, however, a number of Web services are delivered with Siebel CRM. For more information, see Filter Definition Files Available with Siebel CRM on page 112.

  The utilities plug-in for Oracle JDeveloper supports the editing and changing of a series of metadata definition files contained within the configuration projects. In addition, you can configure custom validation rules, business service scripts, and specialized user interfaces layouts using Adobe Flex. You can also use Oracle JDeveloper to edit a series of metadata definition files that define the Sales application for the Offline Client for Life Sciences.

Offline Client for Life Sciences User Interface

The user interface of the Offline Client for Life Sciences consists of metadata that defines the specific information and content displayed in a given application page, and the display-layout definition within its component applications, for example, the Sales application. The Offline Client for Life Sciences combines the information content definition and the display-layout definition with data from the local database to display the user interface at run time.

The user interface consists of a series of related containers and definitions:

- **Application.** The top-level container for the Offline Client for Life Sciences application, for example, the Sales application. Each container defines the pages available in the Sales application and the header and footer banners. Each page defined in the Sales application works independently of the other pages. You can define an application with up to three pages for the Offline Client for Life Sciences. The Sales application uses a three-page layout: Home, Today's Plan, and Quick Access. For each container, a series of pages can be defined which can be changed with each other.
- **Pages.** The second-level container that defines where to display different information in a given area of the Offline Client for Life Sciences application. A page can contain a single applet or sections that contain applets. Applets can also be switched between other applets in the same page.
- **Applets.** The third-level container that defines the information displayed in a specific area.
- **Toolbars**. An independent container defining the toolbars and buttons used in all of the other containers in the Offline Client for Life Sciences application.

The separate XML metadata definition files for each container in the Offline Client for Life Sciences define the content of each container level (for example, Home Page, or Contact list). For more information about XML metadata definition files, see Key Configuration Files on page 101.

A series of generic, display-layout templates for each application container level defines how information must be laid out and displayed for multiple definitions of the same type (for example, a list applet). These layout templates are defined as Adobe Flex MXML files. Each layout template contains a series of placeholders. The content in the metadata definition files maps to these placeholders.

Overview of Client Application Object Definitions

The Offline Client for Life Sciences includes a data layer that supports query, create, update, and delete operations on the local database, and a synchronization layer that supports the synchronization of data between the local database and the Siebel CRM database. This data and synchronization layer is defined in the metadata as a series of related application objects in the filter definition files.

About Filter Definition Files

A *filter definition file* is a file containing the metadata definition of the objects.

For Siebel CRM, the files reflect the structure of Oracle's Siebel CRM Web services to which they map and support. The files can have a flat structure or a hierarchical structure. It is important to note that synchronization uses only the flat structure. The hierarchical structure is used to render the UI.

Filter object definitions include attributes used to create tables in the local database and to manage access to the Web services.

Synchronization of data between the Offline Client for Life Sciences and Siebel CRM occurs through Web services. Filter definition files are used to map data objects and access functions from the Web services to the data model of the local database. The data model includes database entities, relationships between entities, fields in the entities, and the data types of the entities among other database related attributes, such as, the table constraints and various database fields. Consequently, the data model defined by the filter definition files is a subset of the data model defined by the Web services.

Filter definition files also serve as the basis for the data access service that the data layer provides to local Offline Client for Life Sciences applications. This data access service encapsulates the complexity of the data access in the data layer and simplifies design and implementation.

For Siebel CRM, each filter object definition includes a flat file that maps to Oracle's Siebel CRM Web services and their structure. Taking *Related Accounts belonging to a Contact* as an example filter, then the *Related Accounts filter* will be used by synchronization, and the *Related Accounts node under Contact* will be used by the UI.

Each application for the Offline Client for Life Sciences has a set of filter definition files that define the objects used in the application and the data stored in the local database. For more information about filter definitions, see Filter Definitions on page 112.

Siebel CRM uses WSDL definitions, which are delivered for preconfigured objects, to integrate with Siebel CRM.

Data Sources and Offline Databases

Each application for the Offline Client for Life Sciences uses a set of filter object definitions that is synchronized with a specific Siebel CRM instance. The combination of filter definition files and a Siebel CRM instance for each application is called a *data source*. The data source is defined as part of the Sync application configuration in the datasources.xml file.

The Sync application uses the data source information to determine which databases and tables to create for the Offline Client for Life Sciences when it first downloads data from the Siebel CRM instance. Applications for the Offline Client for Life Sciences (for example, Sales) use the local database and tables created by the Sync application.

Two database files are created for each data source synchronized by the Sync application as shown in Table 2. There is also one database for the Offline Client for Life Sciences called System.db. This database contains the active data source information and the connection names that allow the Sync application to communicate with the other applications for the Offline Client for Life Sciences.

Table 2 shows the Siebel CRM database files created for each data source synchronized by the Sync application.

Table 2. Siebel CRM Database Files for the Offline Client for Life Sciences

Database File Name	Description
Sales (CRM On Premise).db. For example: Sales(OfflineClient2).db.	Contains transaction data for your Siebel CRM instance.
Sales (CRM On Premise)Blob.db. For example: Sales(OfflineClient2)Blob.db.	Contains BLOB (binary large objects) data for your Siebel CRM instance.

Overview of Client Data Synchronization

The Offline Client for Life Sciences exchanges data with the Siebel CRM Server using Web services published by that server. The data synchronization is bidirectional:

- The Offline Client for Life Sciences downloads data from the Siebel CRM Server.
- The Offline Client for Life Sciences uploads locally changed data to the Siebel CRM Server.

The data synchronization is automatically started from the Offline Client for Life Sciences.

About Network Connectivity and Automated Synchronization

The synchronization application detects automatically when there is network connectivity and determines when a connection to the Siebel CRM Server is possible. The synchronization application icon changes to a disconnected state when network connectivity cannot be detected. When in a disconnected state, the synchronization application icon is grayed out with a white X mark in a red circle.

The synchronization application automatically synchronizes, according to the assigned synchronization interval, with Siebel CRM when connectivity is available. The synchronization interval is set to 60 minutes by default, which means that synchronization occurs automatically every 60 minutes if the synchronization application can connect to the server. If the synchronization application cannot connect to the server at the synchronization interval, then synchronization will occur when connectivity is next established.

About Synchronization Failure

Synchronization between the Offline Client for Life Sciences and the Siebel CRM Server can fail for a number of different client-side and server-side reasons.

Synchronization Failure Due to Client Issues

The synchronization process can fail for the following client-side reasons, where everything on the server-side is ok:

- There is a VPN failure.
  - If there is a VPN failure during the synchronization process, then the Sync application logs an *Unknown Error and failure message*.

    NOTE: If anything fails during a Full Download, then everything fails. If an attachment download or some part of an attachment during a download fails, then it is ignored and the synchronization download process continues with the next object.
  - If there is a VPN failure when the Sync application is in idle mode, then network connectivity registers as offline.
- There is a system-forced shut down during the synchronization process.

  In this case, the synchronization process is abandoned during an incomplete state. When the Sync application is restarted, it attempts to start a new synchronization process and that new process behaves in the usual way.
- Connection to the server fails during the synchronization process.

  In this case, the Sync application logs an *Unknown Error and failure message*.

Synchronization Failure Due to Server Issues

The synchronization process can fail for the following server-side reasons, where everything on the client-side is ok.

- The Siebel CRM Server shuts down.

  If the server shuts down during the synchronization process, then the Sync application logs an *Unknown Error and failure message*.
- The Siebel CRM Web service has not been published.

  If the Web service has not been published, then the Sync application receives an error from the server saying the Web service does not exist and logs a failure message.
- The computer fails.

  If the computer fails during the synchronization process, then the Sync application logs an *Unknown Error and failure message*.

Data Download Modes

There are three different download modes:

- Full download
- Object download
- Incremental download

Full Download Mode

This mode is selected by the Sync application in specific situations. During a *full download*, all data for the Offline Client for Life Sciences is downloaded, existing local databases are backed up, and the new local databases are built containing the modified data from the server.

The synchronization application uses this mode automatically if at least one of the following conditions is true:

- A new installation is present; that is, no local databases exist.
- Any of the following occurs:
  - The database schema changes.
  - The LOV seed data is different.
  - The datasources.xml file is different.
  - The filter definition file version is different, or if the data schema changes.
- There is an entry in the modification tracking table that signals a full download for a user. For example, after a mass update because of a territory realignment.

  In Siebel CRM, administrators can initiate a full download for a user by creating a new record in the modification tracking table as follows:

  - By navigating to the Administration - Modification Tracking screen in the Siebel application.
  - Clicking the Full Download button.
  - Entering the user information for the new record that is created.

NOTE: The sequence of events during a full download is as follows: first the user is authenticated, then the picklist data is downloaded, followed by the object data, and finally the attachments.

Object Download Mode

This mode is set during the configuration. In an object download, all records for a specific object are downloaded. The object download is triggered either by one of the following:

- If the following entry occurs in the configuration file, datasources.xml, where the value o represents object download.

  ```
  syncType="o"
  ```

- The stale calculation in the synchronization application.

  NOTE: The calculation is not published.

Incremental Download Mode

This mode is set during the configuration. An *incremental download* refreshes the records in the local database that were modified on the server only after the last successful download.

In Siebel CRM, the Object mapping table in the Siebel database records the mapping between Siebel tables and objects belonging to the Offline Client for Life Sciences. As changes are made in Siebel CRM, a modification tracking record is written to the Modification Tracking table to identify the type

of object, the record ID, and type of change that occurred. If an object mapping exists and changes are written to the table, then incremental synchronization for that object can take place provided that the object is marked for incremental download.

During an incremental synchronization, the Sync application does the following:

- Downloads the modification records modified since the last synchronization.
- Evaluates the downloaded modification tracking entries to create a list of records to download for each filter definition file that is marked in the datasources.xml file for an incremental download (that is, syncType="i"). It also accounts for whether the objects can be considered stale.

  An object is *stale* if the time interval since the last successful download of a specific object exceeds the lifetime specified for that object in the datasources.xml file with the lifetime attribute:

```
<ListOfEntitySyncRts>
   <SYS_EntitySyncRt id="Account" ... lifetime="72" ...>40</SYS_EntitySyncRt>
</ListOfEntitySyncRts>
```

  In this example, Lifetime="72" means the Account object is considered stale if there has been no successful download of Account data for more than 72 hours.

  Objects that are stale are downloaded in object download mode, even if they are marked in the datasources.xml file for an incremental download.

Data Upload Modes

The Offline Client for Life Sciences records each commit operation made in the user interface to a dedicated SyncHash table in the local database. When the Sync application starts a synchronization operation, it attempts to upload all the modifications that are registered in the SyncHash table to the server. It distinguishes between the two following cases:

- **Objects with server proprietorship.** These are objects that do not have a single owner, but are shared between users, for example, account objects. For these objects, the Offline Client for Life Sciences performs a query on the server to verify that the data is not modified on the Siebel CRM Server after being downloaded:
  - If the data has been modified on the server, a conflict notice is logged, and the corresponding entry in the SyncHash table container is saved to the log file. During the next synchronization cycle, the Sync application tries again to upload that record to the server. Conflicts must be resolved manually. It is recommended that you decide which data is considered to be valid.
  - If the data on the server has not been modified, the data modified in the Offline Client for Life Sciences is uploaded to the server and applied to the server database.
- **Objects with client proprietorship.** These are objects that are owned by a single user, for example, an activity. The modified data is not checked for records with conflicting changes. The modified data on the Offline Client for Life Sciences is uploaded to the Siebel CRM Server. The modified data overwrites any conflicting changes on the server.

3 Installing the Required Software for the Configuration Environment

This chapter describes how to install and set up the required software for the configuration environment for the Offline Client for Life Sciences. It includes the following topics:

Setting Up the Basic Configuration Environment

To set up Oracle JDeveloper, you must perform the following tasks:

1. Install the required software.

   This includes Adobe AIR 2.5, which is required to install and run the Offline Client for Life Sciences. For more information about all required software and recommended system requirements for the Offline Client for Life Sciences, see *Siebel System Requirements and Supported Platforms* on Oracle Technology Network.

2. Install the required Adobe Flex and Air Software Development Kits (SDKs).

   For more information, see Process of Installing Adobe Flex and Air SDKs on page 20.

CAUTION: When installing the required software or Oracle JDeveloper, make sure that no file directories contain spaces because spaces prevent your installations from working correctly. For example, do not use `D:\My Documents\CONFIG_PROJECTS\appcfg_sales\src\config`. The correct format for this path is: `D:\My_Documents\CONFIG_PROJECTS\appcfg_sales\src\config`.

Process of Installing Adobe Flex and Air SDKs

Adobe Flex and Adobe AIR SDKs (Software Development Kits) must be set up on your computer so you can configure the Offline Client for Life Sciences. A single SDK folder must be set up to contain the combination of the following Adobe SDKs, installed in the following order: Adobe Flex SDK 3.4.1, then Adobe Flex 3.4 Data Visualization Components, and finally Adobe AIR 2.0 SDK.

To install Adobe SDKs, you must perform the following tasks:

1. Locating Adobe Flex SDKs on page 20
2. Installing Adobe Flex SDK 3.4.1 on page 20
3. Installing Adobe Flex SDK 3.4.0 Data Visualization Files on page 21
4. Installing Adobe AIR SDK on page 21

Locating Adobe Flex SDKs

This task is a step in Process of Installing Adobe Flex and Air SDKs on page 20. Depending on your installation of Adobe Flex, the SDK can be placed in the following locations:

- **Adobe Flex with Adobe Flash Builder 4.** If Adobe Flash Builder 4 is installed, you can find its corresponding SDKs under the main Adobe Flex program folder, for example:

  ```
  Program Files\Adobe\Flash Builder 4\sdks
  ```

- If Adobe Flash Builder 4 is included in your installation, you must create a new 3.4.1 folder under the Flash Builder 4\sdks folder, for example:

  ```
  Adobe Flash Builder 4\sdks\3.4.1
  ```

- **Adobe Flex without Adobe Flash Builder 4.** You can configure the Offline Client for Life Sciences metadata without having Flash Builder 4 installed using only Oracle JDeveloper. However, you require the Flex SDK 3.4.1 files to build the application for the Offline Client for Life Sciences, and you must save them in CONFIG_PROJECTS folder, for example:

  ```
  CONFIG_PROJECTS\SDK3.4.1
  ```

Installing Adobe Flex SDK 3.4.1

This topic describes how to install Adobe Flex SDK 3.4.1. This task is a step in Process of Installing Adobe Flex and Air SDKs on page 20.

To install Adobe Flex SDK 3.4.1

1. Open a Web browser, and navigate to the following address:

   ```
   http://opensource.adobe.com/wiki/display/flexsdk/Download+Flex+3
   ```

2. Download the required ZIP file, and extract the contents to the 3.4.1 SDK folder.

Installing Adobe Flex SDK 3.4.0 Data Visualization Files

This topic describes how to install Adobe Flex SDK 3.4.0 data visualization files. This task is a step in Process of Installing Adobe Flex and Air SDKs on page 20.

To install Adobe Flex SDK 3.4.0 data visualization files

1. Open a Web browser, and navigate to the following address:

   http://download.macromedia.com/pub/flex/sdk/datavisualization_sdk3.4.zip

2. Download the required ZIP file, and extract the contents of the data visualization ZIP file into the 3.4.0 SDK folder, overwriting any existing files.

Installing Adobe AIR SDK

This topic describes how to install Adobe AIR SDK. This task is a step in Process of Installing Adobe Flex and Air SDKs on page 20.

To install Adobe AIR SDK

1. Open a Web browser, and navigate to the following address:

   http://www.adobe.com/cfusion/entitlement/index.cfm?e=airsdk

2. Create a temporary folder on your computer.
3. Download the Adobe AIR SDK ZIP file.
4. Extract the contents of the AIR SDK to the temporary folder.
5. Move the extracted files to the 3.4.1 SDK folder on your computer, overwriting any existing files.

Unpacking the Configuration Projects

Before you can configure Oracle JDeveloper, you must unpack and save the configuration projects on the computer for the Offline Client for Life Sciences.

To unpack the configuration projects

1. Create a folder called CONFIG_PROJECTS on a local drive of the client computer where you will save the configuration projects, for example, as shown in the following table.

Product	CONFIG_PROJECTS Folder
Siebel CRM	CONFIG_PROJECTS_OP

2 In Oracle JDeveloper, import the utilities extension plug-in for the Offline Client for Life Sciences.

For more information, see Importing Utility Extensions and Other Upgrades for Oracle JDeveloper on page 31.

Table 3 describes the configuration projects available to you.

Table 3. Configuration Projects for Siebel Offline Client for Life Sciences

Projects for Siebel CRM	Description
amp_80x	Contains the resulting Asset Message Planner application.
appcfg_sales_80x	Contains the configuration files for the Sales application.
appcfg_amp_80x	Contains the configuration files for the Asset Message Planner application.
appcfg_sync_80x	Contains the configuration files for the Sync application as well as the filter definition files for the Sales and Asset Message Planner applications.
externalservices	Contains the UI events and business services accessible from the configuration projects.
iconproject	Contains the icons used by the Offline Client for Life Sciences applications.
key	Contains the helper functions for generating encryption keys.
sales	Contains the resulting Sales application.
splashscreen	Contains the splash screens for the Sales and Asset Message Planner applications.
sync	Contains the resulting Sync application.
templatesproject	Contains the design templates for the Application, Page, and Applet objects.

Process of Building the Offline Client for Life Sciences with Automated Scripts

The configuration projects for the Offline Client for Life Sciences include a set of scripts to automatically build the AIR install files for a configured version. To use the build scripts, you must perform the following tasks:

1. Setting Up the Unzip Program on page 23
2. Verifying the Perl Installation on page 24
3. Configuring the FLEXPATH File Location for Adobe Flex SDK 3.4.1 on page 24
4. Building an Application on page 24
5. Verifying an Application on page 26

Setting Up the Unzip Program

You must install the info-zip.org (Unzip) program on your computer so that the automated build scripts can perform correctly. You can download the info-zip.org (Unzip) program from the following Web site:

http://sourceforge.net

NOTE: If you cannot find the unzip executable on the SourceForge Web site, then go to the following URL address to get the unz600xn.exe file: ftp://ftp.info-zip.org/pub/infozip/win32/.

This task is a step in Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

To set up the Unzip program

1. Create a new UNZIP folder in the CONFIG_PROJECTS folder on your computer.
2. Download the UnZip.zip file for your operating system, for example, unzip60.zip.
3. Extract the files to the new UNZIP folder.
4. Edit the PATH statement so that the UNZIP folder is appended to the end of file name, for example:

   ```
   C:\WINDOWS\system32\WindowsPowerShell\v1.0;D:\MyDocuments\SOnGO\CONFIG_PROJECTS\UNZIP
   ```

   For more information on the editing environment variables, see the online help for your operating system.

Verifying the Perl Installation

The automated build scripts require Perl to be installed on your computer. This task is a step in Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

To verify the Perl installation

1. Open a command line.
2. Type either perl -v (version flag) or perl -h (help flag), and press Enter.

   If the command is not recognized, you must download and install Perl, which is available from the following Web site:

   http://www.perl.org/

Configuring the FLEXPATH File Location for Adobe Flex SDK 3.4.1

You must update the automated build scripts with the local path of the Flex 3.4.1 SDK. This task is a step in Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

To update the automated build scripts

1. Open the buildscripts folder under the CONFIG_PROJECTS folder on your computer.
2. Right-click the runBuild.bat file, and click Edit.
3. Change the path for FLEXPATH to match the full path to the Flex SDK 3.4.1 folder on the local computer, for example, specify the following path:

   ```
   FLEXSDKPATH=D:/CONFIG_PROJECTS/SDK3.4.1
   ```

   NOTE: You must use a slash mark (/) when editing this path.

Building an Application

When you have completed all the set up tasks, build the application in a test environment. You can build an application from the command line or from Oracle JDeveloper (the preferred method). Complete the following procedure to build an application from the command line. This task is a step in Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

Complete the following procedure to build an application from the command line.

To build an application from the command line

1. Start a command line.
2. Navigate to the buildscripts folder under the appropriate directory:

   ```
   CONFIG_PROJECTS
   ```

3 Enter the following command:

```
runBuild %CONFIG_PROJECTS_PATH% %VERSION_NUMBER%
```

where:

- *CONFIG_PROJECTS_PATH* is the full path to the CONFIG_PROJECTS folder. This path must contain slash marks (/).
- *VERSION_NUMBER* is the name or number of the build created. It can be any number or text value.

In Microsoft Windows, you must use a forward slash mark (/), for example, as follows:

```
runBuild D:/CONFIG_PROJECTS 1.1.0
```

The build process starts, and displays the following messages when it successfully finishes:

```
Application sync is now ready

Application sales is now ready

Application amp is now ready
```

The following AIR files are created under the CONFIG_PROJECTS folder:

- Sales.air
- Sync.air
- Amp.air

Complete the following procedure to build an application using Oracle JDeveloper.

To build an application using Oracle JDeveloper

1 Start Oracle JDeveloper.

2 Click Tools, select CRM Offline Client Utilities, and then Build.

3 Select Build and Package. The Build radio button is selected by default.

4 From the Select Application dropdown list:

a Select the required option and then click OK.

For Siebel CRM, select All_OP.

NOTE: If the Select Application dropdown list is empty, click Refresh to reload the data.

b Click OK.

5 Oracle JDeveloper starts the build process for the following applications as required:

- Sales.air
- Sync.air
- Amp.air

An example of the songoconfiguration.xml file follows.

```
<APPLICATIONS>
    <ALL_OD>buildorder_od_all.txt</ALL_OD>
    <SALES>buildorder_od_sales.txt</SALES>
    <AMP>buildorder_od_amp.txt</AMP>
    <SYNC>buildorder_od_sync.txt</SYNC>
    <ALL_OP>buildorder_op_all.txt</ALL_OP>
    <ORACLECRMSALES80X>buildorder_op_sales.txt</ORACLECRMSALES80X>
    <ORACLECRMAMP80X>buildorder_op_amp.txt</ORACLECRMAMP80X>
    <ORACLECRMSYNC80X>buildorder_op_sync.txt</ORACLECRMSYNC80X>
</APPLICATIONS>
```

Notice that each node contains the build order file for that application. If the node does not have a correct setting, the build will not take place.

Verifying an Application

It is recommended that you verify an application before using it. This task is a step in Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

To verify an application before using it

1. Install the AIR files on your computer.
2. Perform a full synchronization with Siebel CRM.

   For more information about installing the applications and synchronizing them with Siebel CRM, see *Getting Started with Siebel Offline Client for Life Sciences*.

Configuring the Offline Client for Life Sciences

This topic describes the configuration procedures which you must perform to start using Oracle JDeveloper:

- Changing the URL for the Offline Client for Life Sciences on page 26
- Hiding User Preferences on page 27

Changing the URL for the Offline Client for Life Sciences

To synchronize the Offline Client for Life Sciences with a Siebel CRM instance, you must configure the URL for that instance in the datasources.xml file. You can use Oracle JDeveloper or any other editor to do this.

To change the URL for the Offline Client for Life Sciences

1 Open the datasources.xml file, for example, from the location shown in the following table.

Product	datasources.xml File Location
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

2 Find the data source section for the required application.

3 Find the <LoginURL> element.

4 Replace the URL value with the short URL value for the Siebel CRM instance.

5 Save the file.

6 Rebuild the Sync, Sales, and Asset Message Planner applications.

Hiding User Preferences

The applications for the Offline Client for Life Sciences have a number of user preferences which a user can set from within the application in the Settings window. You can configure the user preferences not to be displayed in the Settings windows if you want to set all preferences to their default setting. This setting prevents users from changing the preferences.

To hide user preferences

1 Open the userpreferences.xml file, for example, from the location shown in the following table.

Product	userpreferences.xml File Location
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

2 Find the user preference entry that you want to change.

3 Verify that the value attribute is set to the correct default preference value.

4 Change the hide attribute value to True.

5 Repeat Step 2 to Step 4 for the other user preferences in the file.

6 Save the userpreferences.xml file.

7 Rebuild the Sync, Sales, and Asset Message Planner applications.

4 Installing and Setting Up Oracle JDeveloper and Adobe Flash Builder

This chapter describes how to install and configure Oracle JDeveloper and how to set up Adobe Flash Builder. It includes the following topics:

Installing Oracle JDeveloper

This topic describes how to install Oracle JDeveloper 11*g* (jdevstudio11113install.exe). The Studio Edition for Microsoft Windows is the recommended installation, which is the complete edition of Oracle JDeveloper.

NOTE: You only need the basic XML features to configure the Offline Client for Life Sciences.

Before installing Oracle JDeveloper, you must install the required software for the Offline Client for Life Sciences. For more information, see Chapter 3, "Installing the Required Software for the Configuration Environment."

To install Oracle JDeveloper

1. Download Oracle JDeveloper 11g (11.1.1.3.0) from the following OTN location:

   http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html

2. Double-click the jdevstudio11113install.exe file and on the installer Welcome screen, click Next to begin the installation.
3. Choose a Middleware Home directory.

   You can create a new Middleware Home directory, or select one from a list of existing Middleware Home directories. If you choose to create a new Middleware Home directory, the default provided is `C:\Oracle\Middleware\` for Windows, and `$HOME/Oracle/Middleware` for Linux and UNIX platforms.
4. Choose an installation type.

   Selecting Custom takes you to the Choose Products and Components screen, where you can select the components you want to install.
5. Choose the products and components (this applies to custom installations only).

   Select the components that you want to install. The Offline Client for Life Sciences does not require the Oracle WebLogic Server.

6 Select the JDK location.

If required, select the location of your JDK by navigating to the JDK directory that contains the child folder bin, which in turn contains java.exe. For example, select `C:\JDK\jdk1.6.0_11` if the path to java.exe is `C:\JDK\jdk1.6.0_11\bin\jave.exe`.

7 Confirm the product installation directories.

Review the location where the components will be installed. To make changes, click Back and navigate to the required screen. Click Next to continue with the installation.

8 Click Next, and then Next again.

NOTE: It is not necessary to install the Windows Service or choose a shortcut location.

The Installation Summary screen displays the components that will be installed and the total disk space that will be utilized.

The Installation Status displays the progress of the installation.

9 When the Installation Complete screen displays, the installation process has ended. Then do the following:

- Select Run Quickstart to launch installed components and access online documentation.
- Click Done to end the installation process.

The Oracle JDeveloper application is installed.

10 Click Finish when the Installation Completed screen is displayed. Oracle JDeveloper starts automatically.

Uninstalling Oracle JDeveloper

To uninstall Oracle JDeveloper, complete the following procedure.

To uninstall Oracle JDeveloper

1 In Microsoft Windows, open the Control Panel.

2 Click Add or Remove Programs.

3 In the list of programs, select Oracle Fusion Middleware 11.1.1.3.0, and click Remove.

Starting Oracle JDeveloper on Windows

To start Oracle JDeveloper on Windows, complete the following procedure.

To start Oracle JDeveloper on Windows

1. From the Start menu, select All Programs, Oracle Fusion Middleware 11.1.1.x.x, and then JDeveloper Studio 11.1.1.x.x.
2. On the Select Role screen, select the Java Edition option and then click OK.
3. Load the project that you want.

   Click the Application Menu button in the Application Navigator window, select Open Project, and then browse to the location of your project file, ConfigureCRMOfflineClient.jpr. For example:

Product	Project File Location
Siebel CRM	`CONFIG_PROJECTS_OP\appcfg_sales_80x\src\config\ConfigureCRMOfflineClient.jpr`

Process of Setting Up Oracle JDeveloper

To set up Oracle JDeveloper, complete the following tasks:

1. Importing Utility Extensions and Other Upgrades for Oracle JDeveloper on page 31
2. Setting User Preferences for Oracle JDeveloper on page 32

Importing Utility Extensions and Other Upgrades for Oracle JDeveloper

Check for internal utility extensions and other upgrades. Use the following procedure to import internal extensions and other upgrades for Oracle JDeveloper. This task is a step in Process of Setting Up Oracle JDeveloper on page 31.

To import utility extensions and other upgrades for Oracle JDeveloper

1. Start Oracle JDeveloper.
2. Click Help, and then select Check for Updates.
3. In Step 1 of the Check for Updates wizard, click Next and, if prompted, set your proxy settings as follows:
   a. Select the Use HTTP Proxy Server checkbox.
   b. Enter your proxy setting in the Host Name field.
   c. Enter 80 in the Port Number field.
   d. Select the Proxy Server Requires Authentication checkbox.

e Enter your Single Sign On User Name and password.

f Click OK.

4 In Step 2 of the Check for Updates wizard, select the following checkboxes under Search Update Centers, and then click Next:

- Oracle Fusion Middleware Products
- Official Oracle Extension and Updates
- Internal Automatic Updates

5 In Step 3 of the Check for Updates wizard:

a Select the Show Upgrades Only checkbox.

b Search for and select the upgrades that you want to install. For example, you must search for and install the following: *Offline Client for Life Sciences Configuration Tools*.

c Click Next to continue.

6 In step 4 of the Check for Updates wizard, click Finish.

7 When prompted, restart Oracle JDeveloper.

Setting User Preferences for Oracle JDeveloper

You must configure the user settings of Oracle JDeveloper to point to the local configuration environment before you start to use Oracle JDeveloper. This task is a step in Process of Setting Up Oracle JDeveloper on page 31.

To set the user preferences for Oracle JDeveloper

1 Start Oracle JDeveloper.

2 Click Tools, and then select Preferences.

3 On the Preferences screen, select the CRM Offline Client Configuration option, and set the path locations for the following:

- **UI Definition Folder.** Enter the full path to the config subfolder in CONFIG_PROJECTS, for example, as shown in the following table.

Product	Project File Location
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sales_80x\src\config

- **Filter Definition Folder.** Enter the full path to the config subfolder in CONFIG_PROJECTS, for example, as shown in the following table.

Product	Project File Location
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

- **Icons Folder.** Enter the full path to the icons subfolder in CONFIG_PROJECTS, for example, as shown in the following table.

Product	Project File Location
Siebel CRM	CONFIG_PROJECTS_OP\iconproject\icons

- **Templates Folder.** Enter the full path to the templatesproject subfolder in CONFIG_PROJECTS, for example, as shown in the following table.

Product	Project File Location
Siebel CRM	CONFIG_PROJECTS_OP\templatesproject

- **Build Path.** This is where the ActionScript application code exists. The path is used by the build process to find the ActionScript definition files for building applications. An example build path follows:

Product	Project File Location
Siebel CRM	CONFIG_PROJECTS_OP\buildscripts

- **Build Script.** This path is the location where the build scripts exist. For example:

Product	Project File Location
Siebel CRM	CONFIG_PROJECTS_OP

- **Build Version Format.** Enter the version number format in this field. For example: 1.5.dev.020.

For more information about the CONFIG_PROJECTS folder, see Unpacking the Configuration Projects on page 21.

4 Click Tools, select CRM Offline Client Utilities, and then Configure CRM Offline Client.

Oracle JDeveloper loads the project configuration and metadata files.

NOTE: If the Migration Wizard screen displays, click OK in order to migrate the new filter format and project files for the Offline Client for Life Sciences to Oracle JDeveloper.

5 If not already done, edit <LoginURL> in the datasources.xml file to point to the correct Siebel CRM Server.

For more information, see Changing the URL for the Offline Client for Life Sciences on page 26.

6 If required, build the Offline Client for Life Sciences application.

For more information, see Building an Application on page 24.

Process of Setting Up Adobe Flash Builder 4

This topic describes how to set up Adobe Flash Builder 4 for use with Oracle JDeveloper. To set up Adobe Flash Builder 4, perform the following tasks:

1. Importing Configuration Projects for the Offline Client for Life Sciences on page 34
2. Setting the Installed Flex SDK Preference on page 35
3. Setting Workspace Preferences for Adobe Flash Builder on page 35
4. Verifying and Configuring the Build Order on page 36

NOTE: Adobe Flash Builder might not be needed for all configuration or customization scenarios.

Importing Configuration Projects for the Offline Client for Life Sciences

You must import projects from the CONFIG_PROJECTS folder into Adobe Flex Builder. This task is a step in Process of Setting Up Adobe Flash Builder 4 on page 34.

To import the configuration projects into Adobe Flex Builder

1. Open Adobe Flash Builder.
2. From the File menu, choose Import, and then Other.
3. In the Import window, select the General folder, and then Existing Projects into Workspace.
4. Click Next.
5. In the Import window, click Browse and choose the `CONFIG_PROJECTS` folder.
6. Click OK.

   A project list is displayed in the Projects pane.
7. Click Finish.

   The projects from the `CONFIG_PROJECTS` folder are imported into the Adobe Flex Builder workspace.

Setting the Installed Flex SDK Preference

You must specify Adobe Flex SDK 3.4.1 as the SDK to be used in the Adobe Flash Builder 4 Preference settings. This task is a step in Process of Setting Up Adobe Flash Builder 4 on page 34.

To set the installed Flex SDK preference

1. Open Adobe Flash Builder.
2. In the Window menu, click Preferences.
3. Expand the Flex menu item, and click Installed Flex SDKs.
4. Verify that there is an entry for Adobe Flex 3.4 that points to the installed SDK files and that the entry is selected in the check box.
5. If an entry does not exist:
   a. Click Add.
   b. In the Flex SDK location, enter the path to the Flex 3.4.0 SDK folder, for example:

      ```
      C:\Program Files\Adobe\Flash Builder 4\sdks\3.4.1
      ```

   c. In the Flex SDK name, enter a name for the Flex SDK, for example, Flex 3.4.1.
6. Click OK.

Setting Workspace Preferences for Adobe Flash Builder

You must verify that the Adobe Flex Builder workspace points to the CONFIG_PROJECTS folder. This task is a step in Process of Setting Up Adobe Flash Builder 4 on page 34.

To set the Workspace preferences for Adobe Flash Builder

1. Open Adobe Flex Builder.
2. In the Window menu, click Preferences.
3. Expand the General item, and then Workspace menu items.
4. Click Linked Resources.
5. Verify that there is an entry for DOCUMENTS, which points to the full path for the CONFIG_PROJECTS folder.
6. If an entry does not exist:
   a. Click New.
   b. In the Name field, enter DOCUMENTS.
   c. In the Path field, enter the full path to the CONFIG_PROJECTS folder, for example:

      ```
      D:\CONFIG_PROJECTS
      ```

7. Click OK.

Verifying and Configuring the Build Order

It is recommended that you verify the build order before using it. This task is a step in Process of Setting Up Adobe Flash Builder 4 on page 34.

To verify and configure the build order

1. Open Adobe Flash Builder.
2. In the Window menu, click Preferences.
3. Expand the General item, and then Workspace menu items.
4. Click Build Order.
5. Deselect the Use Default Builder Order check box.
6. Change the build order to the following sequence using the Up and Down buttons:
   - Templatesproject
   - Iconproject
   - Key
   - Externalservices
   - SplashScreen
   - Appcfg
   - Appcfg_sales
   - Appcfg_amp
   - Sync
   - Sales
   - Amp

Process of Running the Offline Client for Life Sciences Application Configuration from Adobe Flash Builder

You can run the configuration for the Offline Client for Life Sciences from within Adobe Flash Builder 4 without building the installation files for any Offline Client for Life Sciences applications (for example, the Sales application, and so on). You can run the configuration in this manner to debug custom configuration and scripts from within Adobe Flash Builder 4.

To set up Adobe Flash Builder 4, perform the following tasks:

1. Verifying and Building Projects on page 37
2. Running Offline Client for Life Sciences Applications on page 37

Verifying and Building Projects

Before running the configuration for the Offline Client for Life Sciences, it is recommended that you build and verify the projects so that no warning messages are generated. This task is a step in Process of Running the Offline Client for Life Sciences Application Configuration from Adobe Flash Builder on page 36.

To verify and build a project

1. Open Adobe Flash Builder 4.
2. From the Project menu, click Clean.
3. Select the Clean all projects radio button.
4. Select the Start a Build Immediately check box from the window that is started in Adobe Flash Builder.

   This check box is not available if you have selected Build Automatically in the Project menu.
5. Click OK. The build process starts and completes without generating any warning messages.

Running Offline Client for Life Sciences Applications

Before running any of the Offline Client for Life Sciences applications for the first time, delete the databases, the userpreferences.xml file, and userauthentication.xml file from the Local Store folder of the Sync application. You must run the Sync application project first, followed by one of the other Offline Client for Life Sciences applications (for example, the Sales or Asset Message Planner application). This task is a step in Process of Running the Offline Client for Life Sciences Application Configuration from Adobe Flash Builder on page 36.

To run the Offline Client for Life Sciences applications

1. Select the sync project in the Flex navigator pane.
2. On the toolbar, click the Run icon.

   The Sync application starts and builds the database in the Local Store folder.
3. Verify that the local databases have been created in the Local Store folder of the Sync application.
4. Click the Sync icon in the Microsoft Windows system tray to open the Sync application user interface.
5. In the Sync application, click the Sync button to start a manual synchronization with the data source.
6. Select the sales project in the Adobe Flex navigator pane.
7. On the toolbar, click the Run button.

   The Sales application starts.

5 Getting Started with Oracle JDeveloper

This chapter provides an overview of Oracle JDeveloper 11*g*. It describes how to start using the metadata definition files provided in Oracle JDeveloper to examine aspects of the Sales application interface. This chapter includes the following topics:

Overview of Tasks Using Oracle JDeveloper

Oracle JDeveloper is a stand-alone application, which allows you to perform the following tasks that support editing and configuring metadata definitions:

- **Edit and change the user interface metadata definition.** You use the UI metadata definition files in Oracle JDeveloper to edit and change the metadata definition of the Offline Client for Life Sciences application (such as the Sales application), including the application pages, applets and toolbar configuration.

  For more information, see About Inspecting the Sales Application User Interface Definitions on page 41.

- **Edit and change the filter object metadata definition.** You use the filter definition files in Oracle JDeveloper to modify the metadata definition of the filter object of the Offline Client for Life Sciences application.

  For more information about the filter functionality, see Inspecting Filter Definition Files on page 45.

- **Validate application UI and filter definition files.** You use the validate functionality in Oracle JDeveloper to validate the application UI and filter definition files for an application with a Siebel CRM data source.

  For more information about validation, see the following:

  - Validating Modified Definition Files on page 44
  - Validating Configured Filter Definition Files on page 45

Configuring Oracle JDeveloper for the Offline Client for Life Sciences

Depending on your specific requirements for Oracle JDeveloper, you can perform the following optional configuration task: Changing the Search Specifications for Filter Definition Files.

Changing the Search Specifications for Filter Definition Files

The Offline Client for Life Sciences uses a series of filter definition files to synchronize data with Siebel CRM. Certain filter definition files have predefined search specifications to limit the data downloaded to the Offline Client for Life Sciences. Complete the following procedure to change these search specifications, and to refine the data download.

To change the search specifications on filter definition files

1. Start Oracle JDeveloper.
2. Click File, Open, browse to the location of the activityfilter.xml file as shown, for example, in the following table and then click Open.

Product	Location of activityfilter.xml File
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

3. In the activityfilter.xml Structure window, expand Filters, then Filter, and then EntityList.
4. Click the Entity - Activity node to display the attributes.
5. In the Property Inspector window, edit the SearchSpec attribute value.
6. Save the filter definition file.
7. Rebuild the Sync, Sales, and Asset Message Planner applications.

   For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

About Inspecting the Sales Application User Interface Definitions

You can use Oracle JDeveloper to review and edit the Offline Client for Life Sciences application UI definitions and their relationships.

The Sales application user interface is defined in a series of UI metadata definition files. You can edit the metadata definition files by opening them in Oracle JDeveloper. The Structure window in Oracle JDeveloper displays all the nodes associated with a definition file.

There are four types of UI metadata definition files:

- **Toolbar definition (toolbar_buttons.xml).** Defines the toolbars and buttons used across the entire application.

  For more information, see Inspecting the Toolbar Definition in the Sales Application on page 41.

- **Application definition (applicationdefn.xml).** Defines the page containers and the associated pages displayed within an application. The Sales application uses a three-page layout: Home, Today's Plan, and Quick Access. For each page container, you can define a series of pages that you can switch between.

  For more information, see Inspecting the Sales Application Definition on page 42.

- **Pages definition (pagesdefn.xml).** Defines which applets are available in a page, and which applets can be switched with other applets in the same page.

  For more information, see Inspecting How Pages Are Defined in the Sales Application on page 43.

- **Applets definition (appletsdefn.xml).** Defines the applets used in the entire application.

  For more information, see Inspecting How Applets Are Defined in the Sales Application on page 44.

After you edit a definition file, you must validate that modified definition file. For more information, see Validating Modified Definition Files on page 44.

Inspecting the Toolbar Definition in the Sales Application

Complete the following procedure to examine how toolbars are defined in the Sales application.

To inspect the Toolbars definition

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click toolbar_buttons.xml.
3. In the toolbar_buttons.xml Structure window, expand the TOOLBARS node, then the TOOLBAR - Category_Nav_Buttons node.

   Three buttons are defined in this toolbar, which correspond to the Home, Call Activity and Detailer tabs in the Sales application:

- BUTTON - Home
- BUTTON - Call Activity
- BUTTON - Detailer

4 Click the BUTTON - Home node.

The Property Inspector window shows the attributes of the node selected in the Structure window. Each button has an ICON attribute with an associated icon item. Each button triggers an event, which is defined in the EVENTNAME attribute. In this case, the EVENTNAME attribute has a value of SHOWPAGE. The EVENTSTATE attribute defines the page to be displayed when the event is triggered.

Inspecting the Sales Application Definition

Complete the following procedure to inspect how the Sales application is defined.

To inspect the Sales application definition

1 Start Oracle JDeveloper.

2 In the Application Navigator window, expand the Projects option, and then click applicationdefn.xml.

3 In the applicationdefn.xml Structure window, expand the APPLICATION - Sales node.

A series of individual TEMPLATE_ITEMS are defined for the Sales application, which includes an item for the Category_Nav_Buttons toolbar. Each TEMPLATE_ITEM has a distinct ITEM_IDENTIFIER that determines where each item is displayed in the application layout.

The Sales application can contain three pages of content:

- TEMPLATE_ITEM - Quick Access Page
- TEMPLATE_ITEM - Today's Plan Page
- TEMPLATE_ITEM - Home Page

Each page container has a distinct ITEM_IDENTIFIER, which indicates in what specific page area it appears in the Sales application.

4 Expand the TEMPLATE_ITEM - Home page node.

Six different pages are in this page container:

- PAGE - Home Page
- PAGE - Contact Page
- PAGE - Account Page
- PAGE - Detailer Page
- PAGE - Detailer Summary Page
- PAGE - Message Plan Page

5 Click the individual page definitions to view their specific attributes.

Inspecting How Pages Are Defined in the Sales Application

Complete the following procedure to inspect how pages are defined in the Sales application.

To inspect how pages are defined in the Sales application

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click pagesdefn.xml.
3. In the pagesdefn.xml Structure window, expand the PAGES node.

   Eight different pages are available: Account Page, Contact Page, Detailer Page, Detailer Summary Page, Home Page, Message Plan Page, Quick Access Page, and Todays Plan Page.

   Each page can contain one or more TEMPLATE sections, and each template section can contain one or more TEMPLATE_ITEMS linking to applets.
4. Expand PAGE- Home Page, then TEMPLATE - P_T_1x1 to see it has an attribute called TEMPLATE_ITEM that is associated with the Calendar Applet.
5. Expand PAGE - Todays Plan Page, then TEMPLATE - P_T_TodaysPlan to see it has:
   - An attribute called TEMPLATE_ITEM that is associated with the Todays Plan Tiled Applet - Scheduled.
   - An attribute called TEMPLATE_ITEM that is associated with the Todays Plan Header Applet.

   Expand TEMPLATE_ITEM - Todays Plan Tiled Applet - Scheduled, then TOGGLEAPPLETS. There is one toggle applet defined in the APPLET attribute, which is called, Todays Plan Tiled Applet - Planned.
6. Expand PAGE - Contact Page, then TEMPLATE- P_T_Accordion to see it has two SECTIONS and two TEMPLATE ITEMS defined.

   The sections can be expanded or compressed within the page. Expand SECTION - ContactDetails and SECTION - ContactCall to see all associated applets.

   Under the SECTION - ContactDetails node:
   - The Contact 360 Accordion Header Applet is the header bar on the Contact detail page.
   - The Contact Form Applet contains a series of applets that correspond to the Detail menu in the Contact page of the Sales application. Expand TEMPLATE_ITEM - Contact Form Applet, then TOGGLEAPPLETS to show the applets.
   - Click TOGGLEAPPLET - Contact License List Applet.

     Notice that the parent applet is the Contact 360 Accordion Header Applet.

Inspecting How Applets Are Defined in the Sales Application

Complete the following procedure to inspect how applets are defined in the Sales application.

To inspect how applets are defined in the Sales application

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click appletsdefn.xml.
3. In the appletsdefn.xml Structure window, expand AppletData, then the applets node.
4. Find and expand the APPLET - Contact Form Applet node.

   The FILTER attribute indicates that the applet is associated with the ACCOUNT_FILTER_OP filter definition file.
5. Expand the TEMPLATE - A_T_Lightweight_Form node.

   The WEB_TEMPLATE name is A_T_Lightweight_Form. This applet definition uses this template. The items listed under this node are the TEMPLATE_ITEMS used to define the applet.
6. Click any item under the TEMPLATE - A_T_Lightweight_Form node.

   An ITEM_IDENTIFIER attribute is associated with each item. This attribute determines to which layout group the item is mapped in the template.

   The COLINDEX, COLSPAN, ROWINDEX and ROWSPAN attributes define where the item is displayed. The COLINDEX and ROWINDEX values start at zero (0). The COLSPAN and ROWSPAN values start at 1.

Validating Modified Definition Files

If you edit an existing UI definition file or add a new applet node or page node to a definition file, then you must validate that modified definition file so that all attributes for the node, including the missing attributes, are included. Only nodes that have values are added by default.

Complete the following procedure to validate modified definition files using Oracle JDeveloper for the Sales application.

To validate a modified definition file in Oracle JDeveloper

1. In the Application Navigator window, expand the Projects option, and then select the type of definition file that you want to validate. For example, select appletsdefn.xml, applicationdefn.xml, or pagesdefn.xml.
2. Click Tools, select CRM Offline Client Utilities, and then Validate.
3. After validation, click Refresh in the Application Navigator window to refresh the file.

   A validation error log file called ValidationErrorLog.xml is created recording any errors. Double-click ValidationErrorLog.xml in the Application Navigator window to review the validation errors.

Inspecting Filter Definition Files

You can use Oracle JDeveloper to inspect and edit the application filter definition files. Use the following procedure to inspect the Sales application filter definition files.

To inspect filter definition files in Oracle JDeveloper

1. Start Oracle JDeveloper.
2. Click File, Open, browse to the contactfilter.xml file and then click Open.

Product	Location of contactfilter.xml File
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

3. In the contactfilter.xml Structure window, expand Filters, then Filter, and then EntityList.

   Entity - Contact is the top-level object in this filter definition file and is defined as an entity. A number of fields are defined below the Entity - Contact node. These fields are the active fields defined for the filter definition file.
4. Select the Field - Id node and examine its attributes.
5. Select the Field - LastName node, and examine its attributes.
6. Scroll to the bottom of the Entity - Contact node.

   A number of folder names start with the word *EntityList*. These are the active related item objects defined in the filter definition file. For more information on the different filter attributes, see Filter Definitions on page 112.

Validating Configured Filter Definition Files

You validate configured filter definition files in Oracle JDeveloper to verify that data, for each individual object including the individual fields defined in the filter definition file, can be queried in the Siebel CRM instance. This feature is useful for verifying that the correct objects and access controls have been set up in Siebel CRM for the Offline Client for Life Sciences configuration. When you validate configured filter definition files, you make Web service requests to the Siebel CRM instance. The requests are based on the filter definition files. Any Simple Object Access Protocol (SOAP) errors encountered are displayed in the log file.

To validate a configured filter definition file in Oracle JDeveloper

1 Start Oracle JDeveloper.

2 Click File, Open, browse to the activityfilter.xml file and then click Open.

Product	Location of activityfilter.xml File
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

3 Click Tools, CRM Offline Client Utilities, and then Validate.

4 After validation, click Refresh in the Application Navigator window to refresh the file.

A validation error log file called ValidationErrorLog.xml is created recording any errors. Double-click ValidationErrorLog.xml in the Application Navigator window to review the validation errors.

Changing the Synchronization Interval

The Offline Client for Life Sciences is designed to synchronize with Siebel CRM automatically at defined intervals. This synchronization interval is defined for all data sources in the datasources.xml file and set to 60 minutes by default.

To change the synchronization interval

1 Open the datasources.xml file in the appropriate directory as shown in the following table.

Product	Location of datasources.xml File
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

2 Find the <ListOfDatasources> element at the top of the file.

3 Find the SyncPeriod attribute.

4 Change the value 60 to the synchronization interval that you want.

5 Save the file.

6 Rebuild the Sync, Sales, and Asset Message Planner applications.

For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

Configuring the Timeout Period

If the Siebel CRM Server fails in the middle of a data download, then an appropriate error message is displayed by the Sync application and written to the log file, for example, as follows:

```
"Full Download Failed Killing other workers: Sales (CRM), CONTACT_FILTER_OP
```

By default, the constants.as file sets the timeout period before all URL or SOAP requests time out to 5 minutes. But you can override this default setting by configuring the timeout period before a request times out for each data source node in the datasources.xml file. If the download synchronization process fails for a particular data source after the configured timeout period, then an error message is logged identifying the data source node.

Complete the following procedure to configure the timeout period on a data source in datasources.xml.

To configure the timeout period on a data source in datasources.xml

1 Open the datasources.xml file in the appropriate directory as shown, for example, in the following table.

Product	Location of datasources.xml File
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

2 Find the <SYS_DataSource> element that you want, for example, as shown in the following table.

Product	Location of datasources.xml File
Siebel CRM	<SYS_DataSource id="Sales (CRM On Premise)" applicationLogin="Y" caseSensitive="n" BackupsToKeep="1" timeout="">

3 Change the value of the timeout attribute, for example, as follows:

```
timeout="2"
```

4 Save the file.

5 Rebuild the Sync, Sales, and Asset Message Planner applications.

For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

Changing the Authentication Validity Period

The Offline Client for Life Sciences authenticates the user credentials with Siebel CRM when an initial synchronization occurs. The user's authenticated log-in credentials are stored in an encrypted file, userpreferences.xml, in the Sync application's `Local Store` folder. These authentication credentials are stored and reused for 5 days (the default setting), after which, users are prompted to authenticate their login details again.

Complete the following procedure to change the validity period for the user's authentication credentials.

To change the authentication validity period

1. Open the datasources.xml file in the appropriate directory as shown, for example, in the following table.

Product	Location of datasources.xml File
Siebel CRM	`CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config`

2. Find the datasource section for the required application.
3. Find the <AuthenticationValidity> element.
4. Change the value from 5 to the required number of days.
5. Save the file.
6. Rebuild the Sync, Sales, and Asset Message Planner applications.

   For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

Configuring the Application Startup Timeout Variable

Complete the following procedure to configure the application startup timeout variable, APPCFG_LAUNCH_TIMEOUT. This variable sets the timeout value in seconds before a request to start the Sales or AMP application times out. By default, APPCFG_LAUNCH_TIMEOUT is set to 60.

To configure the application launch timeout variable

1. Start Oracle JDeveloper.
2. Open AppCfg.as in the appropriate directory as shown, for example, in the following table.

Product	AppCfg.as File Location
Siebel CRM	`CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config`

3 Go to the APPCFG_LAUNCH_TIMEOUT variable and set it, for example, as follows:

```
APPCFG_LAUNCH_TIMEOUT=20
```

In this example, application startup requests will timeout after 20 seconds. If you set APPCFG_LAUNCH_TIMEOUT to a value less than 1, then all application startup requests will automatically timeout after 60 seconds.

4 Save the AppCfg.as file.

5 Rebuild the Sync, Sales, and Asset Message Planner applications.

Inspecting Data Sources and Databases

TIP: It is recommended that you use an administrator tool for the SQLite database to open the Offline Client for Life Sciences databases and inspect the table of contents.

The Sync application automatically creates database tables, so you are not required to configure the database tables. However, it can be helpful to examine the contents of the database tables during development and testing. For example, when additional fields or objects are added to the metadata definitions of the application. You can download the following SQLite database administrator tools, and you can use them to examine the contents of your database:

- **SQLite Expert.** This tool is available at:

  http://www.sqliteexpert.com

- **Lita.** This tool is available at:

  http://sqliteadmin.orbmu2k.de

- **SQLite Administrator.** This tool is available at:

  http://sqliteadmin.orbmu2k.de

Use the following procedures to inspect data sources and databases:

- Inspecting Local Database Tables on page 50
- Inspecting the Data Sources Definition File on page 50
- Logging In as a Different User and Rebuilding a Local Database on page 51

Inspecting Local Database Tables

This topic describes how to inspect the local database tables.

To inspect local database tables

1. Make sure that the Offline Client for Life Sciences has performed a full download synchronization.

   For more information about synchronization, see Overview of Client Data Synchronization on page 15.

2. Use the SQLite Administrator tool to open the appropriate Sales database file as shown in the following table.

Product	Sales Database File Name
Siebel CRM	Sales (CRM On Premise).db

3. Inspect the list of database tables.
4. Find and open the Contact table.
5. Inspect the columns in the Contact table.

Inspecting the Data Sources Definition File

This topic describes how to inspect the content of the datasources.xml file and the definition of the top-level entities and filter definition files for a given data source.

To inspect the data sources definition file

1. Open Windows Explorer and navigate to the appropriate folder as shown, for example, in the following table.

Product	Navigate to:
Siebel CRM	`CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config`

2. Find and open the datasources.xml file using an XML editor or Notepad.

   The datasources.xml file defines one or more data sources with which you can perform a synchronization. For each data source definition, the following series of elements exists:

   - **LoginURL.** This element defines the Siebel CRM Server with which you can perform a synchronization. For more information, see the entry on LoginURL in Table 7 on page 105 on SYS_DataSources.
   - **ListOfEntitySyncRts.** This element groups the top level entities or filter definition files to synchronize with, for example, Siebel CRM.

- **EntitySyncRt.** Each of these elements defines a synchronization entity for the data source and maps the filter, XML file, to the Siebel CRM WSDL. For more information, see Table 9 on page 106.

Logging In as a Different User and Rebuilding a Local Database

It might be necessary to log in as a different user and to synchronize with a new, empty local database, or to rebuild the Sync application databases if you no longer require the existing data. To rebuild the databases, you must delete the local databases and userauthentication.xml files.

Logging In as a Different User

To log in to an existing data source as a different user, you delete the existing userauthentication.xml file, and restart the Sync application.

NOTE: Log in as a different user only when the local databases are empty; that is, before data is downloaded from Siebel CRM. Otherwise, the local databases will contain data from two different users or sales representatives.

To log in as a different user

1. Right-click the Sync application icon in the system tray, and click Exit.
2. Navigate to the Local Store folder on your computer for the Sync application.
3. Delete the userauthentication.xml file.
4. Double-click the Sync application icon on your computer desktop to restart the Sync application.
5. Log in to Siebel CRM as a different user, when prompted.

   The Sync application creates a new userauthentication.xml file for the Sync application.

Switching to Different Data Sources

To switch to a different data source, use the Datasource list in the Sync application, and resynchronize the Sync application with the new Siebel CRM instance.

To switch to a different data source

1. Click the Sync icon in the system tray to start the Sync console.
2. From the Datasource list, select the new data source.
3. Click Sync to synchronize data with the new data source.
4. Log in to Siebel CRM, when prompted.

Rebuilding the Local Databases

If you want to build a new database or rebuild an existing database for the Offline Client for Life Sciences, complete the following procedure.

NOTE: The existing data in the local databases is deleted after rebuilding the databases.

To rebuild the local databases

1. Right-click the Sync application icon in the system tray, and click Exit.
2. Navigate to the `Local Store` folder on your computer for the Sync application.
3. Delete the two database files, as shown in the following table, for the data source that you want to rebuild in the `Local Store` folder.

Product	Database Files
Siebel CRM	Delete the following Siebel CRM database files: ■ CRM instance.db ■ CRM instanceBlob.db **NOTE:** *CRM instance* is the Siebel CRM data source.

4. Double-click the Sync application icon on your computer desktop to restart the Sync application.

   The Sync application creates a database file.
5. Perform a manual synchronization by clicking the Sync button.
6. Log in to Siebel CRM when prompted.

   The Sync application creates the database files shown in the following table.

Product	Database Files
Siebel CRM	The Sync application creates the following Siebel CRM database files ■ CRM instance.db ■ CRM instanceBlob.db

6 Managing String Translations

This chapter describes how to manage string translations and to import and export string files. It includes the following topics:

Managing String Translations

This topic describes how to use the Translation Tool and how to modify and locate strings. It includes the following information:

Location of String Files

String files are stored in two properties files:

- **lfs.properties.** Contains labels, constants, and error messages
- **help.properties.** Contains help text

Both files reside in a language folder with a three-letter language code (such as *enu* for English). Only the files in the enu folder contain the complete set of strings for the product. You can populate the files (with the same names) in other language folders to translate the Sales application into the languages that you require. For more information on translating string files, see Translating Strings on page 56.

In addition, the description.lfs.properties file contains optional descriptions for the strings defined in lfs.properties. Translators might use this file to understand the usage and context of the strings to aid them in the translation process.

String Types and Formats

This topic describes the string types and formats for lfs.properties, help.properties, and description.lfs.properties files.

lfs.properties File

The lfs.properties file is a text file that contains one string on each line. Each string must take the following format:

ID=Text

where:

- *ID* has the syntax of IDS_*Source_Identifier*
- *Source* is one of the following:
  - **SONGO.** Indicates that the string is defined and used in the Sales application.
  - **DAEMON.** Indicates that the string is defined and used in the Sync application.
  - **SYNC.** Indicates that the string is defined and used in the Sync application.
  - **AMP.** Indicates that the string is defined and used in the Asset Message Planner application.
  - **UTILITIES.** First defined and used in Oracle JDeveloper.
- *Identifier* is a unique string. It must not contain spaces.
- *Text* is the displayed string. It can contain spaces especially for error messages.

  The example might read as follows: *IDS_DAEMON_A_DOWNLOAD_COMPLETE=Attachment download complete.*

help.properties File

The help.properties file contains help messages, which take the following format:

ID=Text

where:

- *ID* has the syntax of *AppletName_SequenceNumber=text*

  where:
  - *AppletName* is the name of the applet for which the help text is intended.
  - *SequenceNumber* is a number to uniquely identify a help paragraph and the order of the paragraph in the help text
- *Text* is the help string.

NOTE: Both *ID* and *Text* can contain spaces. An example might read as follows: *Calendar Applet_1=Drag and drop contacts to create an appointment with the contact.*

description.lfs.properties File

The description.lfs.properties file contains descriptions, which take the following format:

ID=Description

where:

- *ID* is the same format as in the lfs.properties file.
- *Description* is the explanation of the string.

  The example might read as follows: *IDS_DAEMON_A_DOWNLOAD_COMPLETE=Downloads are complete.*

Opening a String File

This topic describes how to open a string file using the Translation Tool so that you can modify the customer-visible text in any of the supported languages.

To open a string file

1. Start Oracle JDeveloper.
2. Navigate to the language folder (for example, enu) at the location shown in the following table.

Product	contactfilter.xml File Location
Siebel CRM	CONFIG_PROJECTS_OD\appcfg_sync_80x\src\config\

3. Select and open the description.lfs.properties file.

Adding English Strings

Complete the following procedure to add an English string.

To add an English string

1. Start Oracle JDeveloper.
2. Open the lfs.properties file in the following directory:

Product	contactfilter.xml File Location
Siebel CRM	CONFIG_PROJECTS_OD\appcfg_sync_80x\src\config\enu

3 Add an editable list to the top of the list.

For more information on the formats used in the lfs.properties file, see lfs.properties File on page 54.

Then do the following:

a Click the prepopulated text field in the String Id and add the new unique identifier.

The String ID cannot have embedded spaces.

b Click the prepopulated text field in the Source field (`enu\lfs.properties`) to add a new string.

c (Optional) Type text in the Description field.

4 (Optional) To delete a string, select that entry, and then click Delete.

You can delete a single line or multiple lines.

5 Click Save to save the lfs.properties file.

The source file is saved to the location from which it is read.

Translating Strings

Complete the following procedure to translate a string. In this procedure, strings in English and Spanish are displayed, as an example.

To translate a string

1 Start Oracle JDeveloper.

2 Open the lfs.properties located, for example, in `appcfg_sync\src\config\enu`.

3 (Optional) Also open the description.lfs.properties folder located in the same folder. This loads the target file, which contains the translated labels.

4 Open the lfs.properties file in the language folder for Spanish located, for example, in the following directory: `appcfg_sync\src\config\spa`.

5 Enter the Spanish strings listed in the following table for the target file (`spa\lfs.properties`) for the string identifiers.

String ID	Source String (English)	Target String (Spanish)
IDS_SONGO_ACCOUNT	Account	Cuenta
IDS_SONGO_ACCOUNTS	Accounts	Cuentas
IDS_SONGO_ADDRESS	Address	Direccion
IDS_SONGO_ADDRESSES	Addresses	Direcciones
IDS_SONGO_BEST_TIMES	Best Call Time	Mejor Hora Para Llamar
IDS_SONGO_CALL_ANALYSIS	Call History	Historia de Llamadas

String ID	Source String (English)	Target String (Spanish)
IDS_SONGO_CELL	Cell	Telefono Celular
IDS_SONGO_CITY	City	Ciudad
IDS_SONGO_CONTACT_PROFILE	Contact Profile	Perfil del Contacto
IDS_SONGO_DESCRIPTION	Description	Descripcion
IDS_SONGO_EMAIL	Email	Correo Electronico
IDS_SONGO_FIRST_NAME	First Name	Nombre
IDS_SONGO_LAST_NAME	Last Name	Apellido
IDS_SONGO_LAST_VISITED	Last Visited	Ultima Visita
IDS_SONGO_LICENSES	Licenses	Licencias
IDS_SONGO_POSTAL_CODE	Postal Code	Codigo Postal
IDS_SONGO_STATE	State	Estado
IDS_SONGO_TYPE	Type	Tipo
IDS_SONGO_WORK	Work	Telefono Oficina

6 Click Save or Save As to save the target file.

Using Search Functionality to Find Strings

You can use the search functionality in Oracle JDeveloper to perform a text-based search for criteria, such as String ID, Source, Description, or Target columns.

To use the search functionality to find strings

1 Select Search, and then Find in Files.

2 Enter a search word or phrase in the Search Text field.

3 Select the Search Path. For example:

- Active Application (ConfigureOfflineClient.jws)
- Active Project (ConfigureOfflineClient.jpr)
- Files Open in Editors Only
- User-Defined Search Path

4 Select the Case Sensitive check box, and any of the other search options as required.

5 Click OK.

Oracle JDeveloper searches for the text, and returns any results in a message dialog, for example, as follows:

```
Looking for: IDS_DAEMON_ADDITIVE_TIME in ConfigureOfflineClient.jws
Found 0 occurrences in 51 file(s)

Looking for IDS_DAEMON_ADDITIVE_TIME in: Files Open in Editors Only
C:\Configuration_OP\Configuration\appcfg_sync_80x\src\config\enu\lfs.properties (1
match)
(1,1): IDS_DAEMON_ADDITIVE_TIME=Additive Time
```

Adding New Languages to the Sales Application

The Language menu for the Sales application is in the General tab in the Setting User Preferences window. U. S. English (enu) is available by default in the Sales application. Only the English (enu) option contains the complete set of available strings.

The following procedure adds Spanish (spa) as a language option in the user preferences settings in the Sales application to display the Spanish file that you created in Translating Strings on page 56.

To add a new language to the Sales application

1. Using Oracle JDeveloper or a text editor, open the preferencedata.xml file located, for example, in the following directory:

   ```
   appcfg_sync\src\config
   ```

2. Find the <languages> node, and add the following code under the node:

   ```
   <language><language code></language>
   ```

   In this example, Spanish is added to the Sales application:

   ```
   <language><spa></language>
   ```

3. Save the file.
4. Build the Sales application.

   For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

5. Run the Sales application.
6. Click Settings on the Sales application banner.
7. Select spa.
8. Click the Save icon.
9. Exit the Sales application.
10. Run the Sales application again.
11. Double-click a contact in the Contact Quick List.
12. Click Contact 360 to see the contact information displayed in Spanish.

Exporting and Importing String Files

The following procedures describe how to export string files from Oracle JDeveloper and how to import string files into Oracle JDeveloper.

To export a string file

- Click File, then select Export.

To import a string file

- Click File, then select Import.

Rebuilding an Application

After you have completed all changes in the Offline Client for Life Sciences application you are customizing, such as the Sales application, rebuild that application. For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

7 Adding and Using Icons

This chapter describes how to add and use icons in the Offline Client for Life Sciences. It includes the following topics:

Adding Icons

This topic describes how to add an icon to the Offline Client for Life Sciences.

To add an icon

1. Create the icon you require and save it in PNG or JPG format.
2. Rename the icon in the following format: *xyz_W*x*H*.png.

   In this format, *xyz* describes the icon, and *W* and *H* are the width and height of the icon in pixels, for example, newactivity_10x20.png.
3. Save a copy of the icon in the appropriate folder, for example, as shown in the following table.

Product	Project File Location
Siebel CRM	CONFIG_PROJECTS_OP\iconproject\icons

4. Edit the IconProject.as file in the iconproject folder to add your new icon to the m_arrIcons array, as follows:
   a. Add the following two statements to the IconProject.as file:

```
[Embed(source="icons/xyz_10x20.png")]
   private var xyz_10x20:Class;
```

   b. Add the following case statement to the getIcon function:

```
case "xyz_10x20":
   m_mapIconFamily[ICON_DEFAULT]     = xyz_10x20;
   m_mapIconFamily[ICON_DOWN]        = xyz_10x20;
   m_mapIconFamily[ICON_UP]          = xyz_10x20;
   m_mapIconFamily[ICON_OVER]        = xyz_10x20;
   m_mapIconFamily[ICON_SDOWN]       = xyz_10x20;
   m_mapIconFamily[ICON_SUP]         = xyz_10x20;
   m_mapIconFamily[ICON_SOVER]       = xyz_10x20;
   m_mapIconFamily[ICON_DISABLED]    = xyz_10x20;
   m_mapIconFamily[ICON_SDISABLED]   = xyz_10x20;
   break;
```

5 Rebuild the Offline Client for Life Sciences application you have modified, for example, the Sales application.

For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

Using Icons

You can use icons as follows in the Offline Client for Life Sciences:

- **On a normal button.** When you define a button in the toolbar_buttons.xml file, specify:

  ICON="*xyz_10x20*"

- **On a toggle button.** When you define a toggle applet in the pagesdefn.xml file, specify:

  ICON="*xyz_10x20*"

- **On an applet banner.** You can use the icon as a static icon on an applet header or banner, such as the Contact 360 Accordion Header. To do this, specify the following items in a TEMPLATE_ITEM in the appletsdefn.xml file:

  DISPLAY_OBJECT="*xyz_10x20*"

  TYPE="Image"

- **Dynamic display of icons with an icon map.** If you require context-sensitive icons, you can configure an icon map for a TEMPLATE_ITEM in the appletsdefn.xml file by setting TYPE to IconMap and adding several of TEMPLATE_ITEM_USER_PROP similar to the following configuration on the Contact 360 Accordion Header:

```
<TEMPLATE_ITEM DISPLAY_OBJECT="" FIELD="MarketPotential" ITEM_IDENTIFIER="PH_CENTER" HEIGHT="40"
INACTIVE="N" NAME="MarketPotential" POPUP_EDIT="N" READ_ONLY="Y" SEQUENCE="1" SORT="" TAB_STOP=""
TEXT_ALIGNMENT="left" TYPE="IconMap" VISIBLE="Y" WIDTH="13" COMMENTS="" SORT_MODE="ASC" GROUP=""
PICK_APPLET="" CALCULATED="N" EXPRESSION="" AVAILABLE="Y" SHOW_IN_LIST="N" DISPLAY_FORMAT=""
REQUIRED="N" ROWINDEX="" ROWSPAN="" COLSPAN="" COLINDEX="" DATA_TYPE="DTYPE_TEXT" NO_COPY="N"
PRE_DEFAULT="" DISPLAY_OBJECT_TEXT="">
  <TEMPLATE_ITEMS_USER_PROPS>
    <TEMPLATE_ITEM_USER_PROP INACTIVE="N" NAME="MapValueToIcon:High" VALUE="rank1_Bar_13x40"/>
    <TEMPLATE_ITEM_USER_PROP INACTIVE="N" NAME="MapValueToIcon:Low" VALUE="rank3_Bar_13x30"/>
    <TEMPLATE_ITEM_USER_PROP INACTIVE="N" NAME="MapValueToIcon:Medium" VALUE="rank2_Bar_13x40"/>
  </TEMPLATE_ITEMS_USER_PROPS>
</TEMPLATE_ITEM>
```

  In this example, rank1_Bar_13x40 is displayed when MarketPotential="High", rank2_Bar_13x30 is displayed when MarketPotential="Low", and so on.

- **On the application banner.** You can define an icon on the application banner by adding the following lines of code to the applicationdefn.xml files:

```
<TEMPLATE_ITEM ITEM_IDENTIFIER="APP_TITLE_LEFT" TYPE="Image"
NAME="oracleLogo_135x24" NAME_TEXT=""/>
```

8 Extending the Basic Content in the Sales Application

This chapter describes how to extend the basic content in the Sales application for Siebel CRM, where fields and objects can be customized using Siebel Tools and made available through integration objects as Web services for use in the Offline Client for Life Sciences. It includes the following topics:

Process of Adding and Making Static Fields Available

This topic describes the process of adding and making static fields available in the Sales application. The following picklist field in the Contact record is used as an example:

Field - MrMrs

To add and make a static field, such as the MrMrs picklist field, available in the Sales application, complete the following tasks as required:

Finding the List of Values Name for the Picklist Field

This task is a step in Process of Adding and Making Static Fields Available on page 63. It is recommended that you locate the List of Values (LOV) for the picklist field using an SQLite database administrator tool. For more information about this tool, see Inspecting Data Sources and Databases on page 49.

To find the LOV name for the picklist field

1. Synchronize the Sales application with Siebel CRM.
2. Use an SQLite Database Administrator Tool to open the local database.
3. Find and open the SYS_LOV table.
4. Sort the LOVType column.
5. Find the LOV type for the required field and note the name as shown, for example, in the following table.

Product	LOV Type for the Field
Siebel CRM	MR_MS

Setting the LOV for the Field Using the Filter Functionality

This task is a step in Process of Adding and Making Static Fields Available on page 63. You can set the LOV on a field using the filter functionality in Oracle JDeveloper.

To set the LOV on the field using the filter functionality

1. Start Oracle JDeveloper.
2. Click File, Open, browse to the location of the contactfilter.xml file as shown, for example, in the following table and then click Open.

Product	contactfilter.xml File Location
Siebel CRM	CONFIG_PROJECTS_OD\appcfg_sync_80x\src\config\

3. In the contactfilter.xml Structure window, expand Filters, Filter, EntityList, then Entity - Contact.
4. Select a field and set the list of value name in Property Inspector using the LOVType attribute, as shown, for example, in the following table.

Product	Field Name	Attribute	Attribute Value
Siebel CRM	Field - MrMrs	LOVType	MR_MS

5. Save the file.
6. Validate the modified definition file.

   For more information, see Validating Configured Filter Definition Files on page 45.

Adding the Field to a Form Applet

This task is a step in Process of Adding and Making Static Fields Available on page 63. After setting the LOV on the field using the filter functionality, you must use the UI builder functionality to add the field to a form applet in the Sales application.

To add the field to a form applet

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click appletsdefn.xml.
3. In the appletsdefn.xml Structure window, expand AppletData, applets, and then APPLET - Contact Form Applet.
4. Right-click A_T_Lightweight_Form_Full_Width_Section1, select Insert inside A_T_Lightweight_Form_Full_Width_Section1, then TEMPLATE_ITEM.
5. Enter the following information in the Insert TEMPLATE_ITEM dialog that displays, then click OK.

Field Name	Value
NAME	Title
TYPE	Combobox?

6. Use Property Inspector to set the following attribute values for TEMPLATE_ITEM - Med Ed Event Page:

Attribute	Value
NAME	Title
FIELD	MrMrs
TYPE	Combobox
COLINDEX	0
COLSPAN	1
ROWINDEX	8
ROWSPAN	1
WIDTH	100
HEIGHT	22
ITEM_IDENTIFIER	PHn
SEQUENCE	*Unique number*
DISPLAY_OBJECT_TEXT	CallFrequency

NOTE: When you click an editable field, picklists, text boxes, and date editors appear automatically depending on the field type.

7 Save the file.

8 Validate the modified definition file.

For more information, see Validating Configured Filter Definition Files on page 45.

Rebuilding the Sync and Sales Applications

This task is a step in Process of Adding and Making Static Fields Available on page 63. After you have made your modifications to the Sales application attributes, you must rebuild and test the Sales application.

To build and test the Sync and Sales applications

1 Build the Sync application and the Sales application.

For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

2 Start the Sync application.

3 Start the Sales application and verify that the changes you have made are operational.

Process of Updating Fields in an Existing Filter Definition File

This topic describes the process for updating a field in the Contact filter definition file and then making that field available in the Contacts page in the Sales application. The Contact filter definition file and the Contacts record in Siebel CRM is used as an example.

To activate fields in a filter definition file for the Sales application, complete the following tasks as required:

1 Adding a New Field on page 67

2 Activating the Field Using the Filter Functionality on page 67

3 Rebuilding the Sync Application on page 68

4 Verifying the LOV Name and LOV Values on page 68

5 Making the New Field Available in a Form on page 69

Adding a New Field

This task is a step in Process of Updating Fields in an Existing Filter Definition File on page 66. Complete the following procedure to add a new field to the Contact record in Siebel CRM.

To add a new Siebel CRM field

1. Log in to Siebel CRM as an administrator.
2. Navigate to the Administration - Data screen, then the List of Values view.
3. Click New to add new list of value records of Type MR_MS with, for example, the display values shown in the following table.

Type	Display Value
MR_MS	Dr
MR_MS	Miss
MR_MS	Mr
MR_MS	Mrs
MR_MS	Mrs

   For information about finding the LOV name, see "Finding the List of Values Name for the Picklist Field" on page 63.

For more information about adding fields in Siebel CRM, see *Siebel Applications Administration Guide* and *Using Siebel Tools* on OTN.

Activating the Field Using the Filter Functionality

This task is a step in Process of Updating Fields in an Existing Filter Definition File on page 66. After verifying the generic name of the field, you must activate the field in the filter definition file for the object using the filter functionality in Oracle JDeveloper.

To activate the field using the filter functionality

1. Start Oracle JDeveloper.
2. Select Edit, then Update Filter, and in the Update Filter File dialog that displays:
   a. Browse to the location of the filter that you want to update.
   b. Browse to the location of the WSDL file, which has the new field that you want to add to the filter file.
3. Click OK and in the dialog that displays, drag and drop the field that you want from the left pane into the right pane.

4 Click Finish.

The filter now has the new field.

5 Save the file.

6 Validate the modified definition file.

For more information, see Validating Configured Filter Definition Files on page 45.

Rebuilding the Sync Application

This task is a step in Process of Updating Fields in an Existing Filter Definition File on page 66. After activating the field, rebuild the Sync application by completing the following procedure.

To rebuild the Sync application

1 Exit the Sales application, if it is running.

2 Exit the Sync application, if it is running.

3 Delete the databases, with extension .db, from the Sync application `Local Store` folder, which are typically found at the location shown in the following table.

Product	Database Location
Siebel CRM	`C:\Documents and Settings\user\Application Data\Sync80x\`

4 Build the Sync application.

For more information about building an application, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

5 Start the Sync application

Building the Sync application recreates the databases.

6 Open the database using the SQLite Database Administrator Tool.

For more information about this tool, see Inspecting Data Sources and Databases on page 49.

7 Verify the new list of values created for MR_MS.

Verifying the LOV Name and LOV Values

This task is a step in Process of Updating Fields in an Existing Filter Definition File on page 66. Because a new picklist field has been added to Siebel CRM, the LOV name must be verified to complete the filter definition file configuration.

To verify the LOV name and LOV values

1. Synchronize the Sync application with Siebel CRM.
2. Using the SQLite Database Administrator Tool, open the transaction database.
3. Find and Open the SYS_LOV table.
4. Find and note the LOV type for the new field. For example: MR_MS.

Making the New Field Available in a Form

This task is a step in Process of Updating Fields in an Existing Filter Definition File on page 66. After verifying the LOV name you must make the new field available in a form, in the Sales application.

To make the new field available in a form

1. Set the LOVType attribute for the new MR_MS field in the Contact filter definition file.

   For more information, see Setting the LOV for the Field Using the Filter Functionality on page 64.

2. Add the field to the Contact form applet.

   For more information, see Adding the Field to a Form Applet on page 65.

Process of Making a New Related Item Available

This topic describes how to make available the MedEd Events associated with a specific physician in the Offline Client for Life Sciences. Note that the MedEd Event object is available in Siebel CRM but not in the Offline Client for Life Sciences Release 1.5; as a result, MedEd Event is treated as a custom object.

To make related items available in the Sales application for Siebel CRM, complete the following tasks:

1. Identifying the Required Siebel CRM Objects on page 70
2. Building Business Services in Siebel Tools for Siebel CRM on page 70
3. Embedding the MedEd Event Custom Object Structure in the Contact Filter Definition File in Siebel CRM on page 71
4. Adding a New List Applet on page 73
5. Adding the Applet to the Contact Page on page 74
6. Rebuilding the Sync and Sales Applications on page 75

Identifying the Required Siebel CRM Objects

Use the following procedure to identify the required Siebel CRM objects. This task is a step in Process of Making a New Related Item Available on page 69.

To identify the required Siebel CRM objects

1. Log in to Siebel CRM as an administrator.
2. Navigate to the Contact screen, then select and open any contact record.
3. Select the MedEd view (or Pharma Professional MedEd View) and make a note of the following:
   - The Business Objects. For example: Contact.
   - The Business Components. For example:
     - Contact
     - Pharma ME Invitee Event
     - Pharma ME Session

Building Business Services in Siebel Tools for Siebel CRM

For Siebel CRM, fields and objects can be customized using Siebel Tools and made available through integration objects as Web services for use in the Offline Client for Life Sciences. Use the following procedure to build a business service in Siebel Tools for Siebel CRM. This task is a step in Process of Making a New Related Item Available on page 69.

To build a business service in Siebel Tools for Siebel CRM

1. Log in to Siebel Tools.
2. Search for the Pharma Me Invitee Event business component.

   Notice that the parent business object for the Pharma ME Invitee Event is Contact. Siebel CRM uses flat filters for synchronization, so you cannot use the Contact business object, you must create a new business object for Pharma ME Invitee Event.
3. Create a new business object with the Pharma ME Invitee Event business component as the only child.

   For more information about creating a new business object, see *Using Siebel Tools* and *Siebel Tools Online Help*.
4. Create a filter called, for example, contactmededfilter.xml.

   For more information, see Roadmap for Creating a New Filter Definition File for a Top-Level Object on page 87. This topic describes how to create a top-level, or parent, object called MedEd Event in Oracle JDeveloper.

Embedding the MedEd Event Custom Object Structure in the Contact Filter Definition File in Siebel CRM

Use the following procedure to embed the MedEd Event custom object structure in the Contact filter definition file using Oracle JDeveloper. This task is a step in Process of Making a New Related Item Available on page 69.

To embed the MedEd Event custom object structure in the Contact filter definition file

1. Start Oracle JDeveloper.
2. Click File, Open, browse to the location of the contactfilter.xml file as shown, for example, in the following table and then click Open.

Product	contactfilter.xml File Location
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

3. In the contactfilter.xml Structure window, expand Filters, Filter, EntityList, and then Entity - Contact.
4. Copy the PharmaMeInviteeEvent EntityList from the contactmededfilter.xml file and paste it as one of the nodes into the contactfilter.xml child entity.
   a. Select Entity - PharmaMeInviteeEvent and in Property Inspector, set the value of the ChildKey attribute to ContactId, as shown in the following example.
   b. Find and select the ContactId field under Entity - PharmaMeInviteeEvent and set the ForeignKeyTo attribute to Contact, as shown in the following example.

   In the following example, the ChildKey is CondactId and the ParentKey is Contact.ContactId. For more information about creating a filter, see Creating a Filter Definition File Using the Filter Functionality on page 90.

```
<EntityList Type="ListOfMedEd_Events" NoSync="N" Inactive="N" Associate="N"
CascadeDelete="N" Insert="N">

   <Entity Type="PharmaMeInviteeEvent" Entity="PharmaMeInviteeEvent"
   CascadeDelete="N" Childkey="ContactId" Parentkey="Contact.ContactId">

      <Field Type="Id" Datatype="DTYPE_TEXT" Entity="PharmaMeInviteeEvent"
      Attribute="Id" RowId="Y" UserKey="0" Filterable="Y" Required="N" ReadOnly="Y"
      NoSync="N" SystemField="Y" Inactive="N"> <Constraint Id="UNIQUE"/>
      <Constraint Id="NOT NULL"/> </Field>

      <Field Type="Created" Datatype="DTYPE_UTCDATETIME"
      Entity="PharmaMeInviteeEvent" Attribute="CreatedDate" SystemField="Y"
      DefaultValue="" RowId="" Required="N" NoSync="N" Filterable="Y"/>

      <Field Type="CreatedBy" Datatype="DTYPE_TEXT" Entity="PharmaMeInviteeEvent"
      Attribute="CreatedById" RowId="N" UserKey="" Filterable="Y" Required="N"
      ReadOnly="Y" NoSync="N" SystemField="Y" NoColumn="N"
      ForeignKeyTo="SYS_CurrentUser"/>
```

```
        <Field Type="Updated" Datatype="DTYPE_UTCDATETIME"
        Entity="PharmaMeInviteeEvent" Attribute="ModifiedDate" RowId="N" UserKey=""
        Filterable="Y" Required="N" ReadOnly="Y" NoSync="N" SystemField="Y"
        NoColumn="N" Inactive="N"/>

        <Field Type="UpdatedBy" Datatype="DTYPE_TEXT" Entity="PharmaMeInviteeEvent"
        Attribute="ModifiedById" RowId="N" UserKey="" Filterable="Y" Required="N"
        ReadOnly="Y" NoSync="N" SystemField="Y" ForeignKeyTo="SYS_CurrentUser"/>

        <Field Type="ContactId" Datatype="DTYPE_TEXT" Entity="PharmaMeInviteeEvent"
        Attribute="ContactId" RowId="N" UserKey="" Filterable="Y" Required="Y"
        ReadOnly="N" NoSync="N" SystemField="N" Inactive="N" ForeignKeyTo="Contact"/>

        <Field Type="EventId" Datatype="DTYPE_TEXT" Entity="PharmaMeInviteeEvent"
        Attribute="EventId" RowId="N" UserKey="" Filterable="Y" Required="N"
        ReadOnly="Y" NoSync="N" SystemField="Y" NoColumn="N" Inactive="N" />

        <Field Type="EventName" Datatype="DTYPE_TEXT" Entity="PharmaMeInviteeEvent"
        Attribute="EventName" RowId="N" UserKey="" Filterable="Y" Required="N"
        ReadOnly="N" NoSync="N" SystemField="N" NoColumn="N" Inactive="N"/>

        <Field Type="Objective" Datatype="DTYPE_TEXT" Entity="PharmaMeInviteeEvent"
        Attribute="Objective" RowId="N" UserKey="" Filterable="N" Required="N"
        ReadOnly="N" NoSync="N" SystemField="Y" NoColumn="N" Inactive="N"/>

    </Entity>

</EntityList>
```

5 Save the Contact filter definition file.

6 Validate the modified definition file.

For more information, see Validating Configured Filter Definition Files on page 45.

Adding a New List Applet

This task is a step in Process of Making a New Related Item Available on page 69. Using the UI builder functionality in Oracle JDeveloper, add an applet for the new list.

To add a new list applet

1 Start Oracle JDeveloper.

2 In the Application Navigator window, expand the Projects option, and then click appletsdefn.xml.

3 In the appletsdefn.xml Structure window, expand AppletData, and then applets.

4 Add a new list applet as follows:

a Right-click Account Address Pick Applet in the Structure window, select Insert before Applet - Account Address Pick Applet, then APPLET, enter the following information in the Insert APPLET dialog that displays, and then click OK.

Field	Value
NAME	Contact Product Segment List
CLASS	ListAppletObj
TYPE	Standard

b Use Property Inspector to set the following attributes for APPLET - Contact Product Segment List:

Attribute	Value
ENTITY	MedEd
FILTER	CONTACT_FILTER_OP
WIDTH	100%
HEIGHT	100%

NOTE: When you click an editable field, picklists, text boxes, and date editors appear automatically depending on the field type.

5 Insert a template within the new list applet, APPLET - Contact Product Segment List, as follows:

a Right-click APPLET - Contact Product Segment List, select Insert inside APPLET - Contact Product Segment List, then TEMPLATE, enter the following information in the Insert TEMPLATE dialog that displays, and then click OK.

Field	Value
WEB_TEMPLATE	A_T_Lightweight_List_Full_Width_Section1
NAME	A_T_Lightweight_List_Full_Width_Section1

b Use Property Inspector to set the following attribute values for TEMPLATE - A_T_Lightweight_List_Full_Width_Section1:

Attribute	Value
SEQUENCE	PHn

6 Add template items to the new template as follows:

a Right-click TEMPLATE - A_T_Lightweight_List_Full_Width_Section1, select Insert inside TEMPLATE - A_T_Lightweight_List_Full_Width_Section1, then TEMPLATE_ITEM.

b Enter the following information in the Insert TEMPLATE_ITEM dialog that displays and then click OK.

Field	Value
NAME	Name
TYPE	TextInput

c Use Property Inspector to set the following attribute values for TEMPLATE_ITEM - ProductStatus:

Attribute	Value
DISPLAY_OBJECT	IDS_SONGO_NAME
FIELD	Name
ITEM_IDENTIFIER	PHn
SEQUENCE	1

7 Click Save.

8 Validate the modified definition file.

For more information, see Validating Configured Filter Definition Files on page 45.

Adding the Applet to the Contact Page

This task is a step in Process of Making a New Related Item Available on page 69. After adding the new list applet you add the applet to the Contact page in the Sales application.

To add the applet to the Contact Page

1 Start Oracle JDeveloper.

2 In the Application Navigator window, expand the Projects option, and then click pagesdefn.xml.

3 In the pagesdefn.xml Structure window, expand the PAGES node, then PAGE - Contact Page, TEMPLATE - P_T_Accordion, and then SECTION - ContactDetails.

4 Find and expand TEMPLATE_ITEM - Contact Form Applet.

5 Right-click TOGGLEAPPLETS, select Insert inside TOGGLEAPPLETS, then TOGGLEAPPLETS, enter the following information in the Insert TOGGLEAPPLETS dialog that displays, then click OK.

Field	Value
APPLET	MedEd List
VISIBLE	True

6 Use Property Inspector to set the following attributes for TOGGLEAPPLET - Contact Product Segment List:

Attribute	Value
PARENT_APPLET	Contact 360 Accordion Header Applet
CAPTION	Product Segments
GROUP	PageToggle1
ICON	tableview_24x24

7 Click Save.

8 Validate the modified definition file.

For more information, see Validating Configured Filter Definition Files on page 45.

Rebuilding the Sync and Sales Applications

This task is a step in Process of Making a New Related Item Available on page 69. After you have modified the Sales application attributes, you must rebuild, and test the Sales application.

To build and test the Sync and Sales applications

1 Build the Sync and the Sales applications.

For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

2 Start the Sync application.

3 Synchronize the Sync application databases with Siebel CRM by clicking the Sync button.

4 Start the Sales application, and verify that the changes you have made are operational.

For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

Adding a Joined-In Field

This topic describes how to add a joined-in field for a product name to an entity in the Contact filter definition file and to make the field available in the related item list.

To add a joined-in field

1. Start Oracle JDeveloper.
2. Click File, Open, browse to the location of the contactfilter.xml file as shown, for example, in the following table and then click Open.

Product	contactfilter.xml File Location
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

3. In the contactfilter.xml Structure window, expand Filters, Filter, EntityList, and then Entity - Contact.
4. Find and expand, for example, Entity - Account.
5. Select the field that you want to configure as a joined-in field, for example, as shown in the following table.

Product	Field
Siebel CRM	AccountStatus

6. Use Property Inspector to set the following attributes for your field:

Field	Attributes	Value
AccountStatus	Entity	Account
	SourceKey	AccountId
	DestinationKey	AccountId
	AttributeDestination	AccountStatus
	NoColumn	true
	ReadOnly	true
	NoSync	true

Field	Attributes	Value
AccountType	Entity	Account
	SourceKey	AccountId
	DestinationKey	AccountId
	AttributeDestination	AccountType
	NoColumn	true
	ReadOnly	true
	NoSync	true

7 Click Save.

8 Add the joined-in field to the list applet:

a Add the new field to the Contact Product Segment List applet.

For more information, see Process of Making a New Related Item Available on page 69.

b Exit the Sync application and Sales application.

c Build the Sync application and Sales application.

For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

d Start the Sync application and Sales application.

For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23 and *Getting Started with Siebel Offline Client for Life Sciences*.

Adding a Drag-and-Drop Association

This topic describes how to add a drag-and-drop association between applets by adding a predefined example.

To add a drag-and-drop association

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click appletsdefn.xml.
3. In the appletsdefn.xml Structure window, expand AppletData, and then:
   a. Find and expand APPLET - Available Samples Tiled Applet. This applet contains a list of the available samples, which can be accessed using the Samples tab on the Contact Call page.

      In Property Inspector, make sure that DRAG_ENABLED is set to true. This attribute sets up drag support on the source applet.

   b. Find and expand APPLET - Samples Drop List Applet. This applet contains a list of the samples dropped that are associated with a call, which can be accessed using the Samples tab.

      - In Property Inspector, make sure that DROP_ENABLED is set to true. This attribute sets up drop support on the target applet.
      - Expand the APPLET_USER_PROPS node, and select the APPLET_USER_PROP - Lookup:SampleInventory user property.

        The VALUE attribute, ProductId, indicates which pickmaps are called when that entity is dropped. In this case, the pickmap specified is for ProductId. A *pickmap* copies a field's value from a source object's record to the current object's record when the source object's record is picked or dropped.
      - Expand the TEMPLATE - A_T_Lightweight_List_Half_Width node, then TEMPLATE_ITEM - ProductID, and then PICKMAPS.

        To set up a drop on an applet, a pickmap must be defined for the Foreign key ID field and also for any joined-in fields displayed in the drop applet. In this case, a pickmap is defined for Product Id and Product Name.
      - Return to APPLET_USER_PROPS, and select the APPLET_USER_PROP - OnDrop:SampleInventory user property.

        The VALUE attribute, Insert, indicates which operation must be triggered when a drag-and-drop action occurs. In this case, an insert operation must be triggered.

        The VALUE attribute setting for the OnDrop:SampleInventory user property can also be an expression if an insert operation and update operation are required. For example, you may want to update a contact record that already exists in the target applet or you may want to create a new one. If so, then use the following expression:

        ```
        Expr:IIF([ContactId]="","Copy","Update")
        ```

9 Extending the Business Functions

This chapter describes how to extend the business functions in the Offline Client for Life Sciences application. This chapter describes how you can extend or override the features in the Sales application by adding business services and invoking them from the Sales application UI. It includes the following topics:

Process of Defining Business Services

This process describes how to replace the current Delete button in the Contact License List Applet in the Sales application with a new custom Delete button, which will prompt a user for confirmation before deleting the record. The current Delete button deletes a license record when it is clicked without the prompt.

To define a business service, complete the following tasks:

Adding a Business Service

This task is a step in Process of Defining Business Services on page 79. Create a new MyServices.as business service file, and configure the ServiceManager.as file so that it recognizes the newly created MyServices.as file as a business service file.

To add and configure a business service

1. Create a new file called MyServices.as, and save it in the following directory:

   ```
   CONFIG_PROJECTS\externalservices\src\services
   ```

2. Cut and paste the following code into the MyServices.as file:

   ```
   package services
   {
      import businessservices.IService;
      import businessservices.ServiceBase;
      // import mx.collections.XMLListCollection;
      import sharedobjs.GlobalObjs;
   ```

```
//import sharedutils.ExprParser;
//import sharedutils.StringHelper;
import mx.controls.Alert;
import mx.events.CloseEvent;
import sharedobjs.UIEvent;
public class MyServices extends ServiceBase implements IService
{
   private var m_arrApplets:Array;
   //
   // Define your business service methods here
   //
   public function LSValidations():void
   {
      m_arrMethods = new Array("MyDelete");
   }
   //
   // Function to invoke a method
   //
   public function InvokeMethod(func:String, param:Object):Object
   {
      try
         {
            var result:Object = new Object();
            result.status = false;
            result.errMsg = func;
            if (func == "MyDelete")
            {
               result = MyDelete();
            }
         }
         catch(error:Error)
         {
            GlobalObjs.AppMsgLog.logMessage(error, "MyServices::InvokeMethod "
            + error.message);
         }
            return result;
   }
   //
   // Function to raise an alert to get user confirmations
   //
   private function MyDelete(data:Object = null):Object
   {
      try
         {
            var result:Object = new Object;
            result.status = true;
            result.errMsg = "MyDelete";

            Alert.show("You are about to delete a record. Click OK to proceed or
            Cancel to abort.", "Delete Confirmation",Alert.OK | Alert.CANCEL,
```

```
                null, alertListener, null, Alert.CANCEL);
            }
            catch(error:Error)
            {
               GlobalObjs.AppMsgLog.logMessage(error, "MyDelete " + error.message);
               result.status = false;
            }
               return result;
         }
         //
         // Function to handle CloseEvent from the alert and raise a DELETEDATA event
            upon positive confirmation from users
         //
         private function alertListener(eventObj:CloseEvent):void
         {
            if (eventObj.detail == Alert.OK)
            {
               var evt:UIEvent = new UIEvent(UIEvent.DELETEDATA);
               evt.directListener = this.applet;
               evt.state = "DEFAULT";
               GlobalObjs.Notification.Notify(evt);
            }
         }
      }
   }
```

3 Save the MyServices.as file.

4 Open the ServiceManager.as file in the following directory:

```
CONFIG_PROJECTS\externalservices\src\services
```

5 Add the following line to the m_arrServiceClasses:

```
"MyServices",
```

6 Define a ClassFactory by adding the following line:

```
private var m_cfMyServices:ClassFactory = new ClassFactory(MyServices);
```

NOTE: A ClassFactory is a factory object that Adobe Flex uses to generate instances of another class, each with identical properties.

7 Add the following line to the GetService() function:

```
case "MyServices":
  cf = m_cfMyServices;
  break;
```

8 Save the ServiceManager.as file.

Adding a Button to Call the Business Service

This task is a step in Process of Defining Business Services on page 79. Create a new button that will call the new business service when it is clicked.

To add a button to call the business service

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click toolbar_buttons.xml.
3. In the toolbar_buttons.xml Structure window, expand the TOOLBARS node.
4. Right-click the TOOLBAR - AppletToolbar - Standard - D node and select Copy.
5. Right-click TOOLBARS and select Paste.
6. In the Property Inspector window, make the following attribute value changes for the new node:
   - Change NAME to My Delete Button
   - Change EVENTNAME to INVOKESERVICE.
   - Change EVENTSTATE to MyServices::MyDelete.
7. Click Save.
8. Validate the modified definition file.

   For more information, see Validating Configured Filter Definition Files on page 45.

Using a Button in an Applet

This task is a step in Process of Defining Business Services on page 79. Replace the current Delete button on the Contact License List Applet with the newly created My Delete button.

To use a button in an applet

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click appletsdefn.xml.
3. In the appletsdefn.xml Structure window, expand AppletData, and then applets.
4. Find and expand APPLET - Contact License List Applet node, then TEMPLATE - A_T_Lightweight_List_Full_Width_Section1, and select TEMPLATE_ITEM - Delete Button.
5. In the Property Inspector window, change DISPLAY_OBJECT to My Delete Button.
6. Click Save.
7. Validate the modified definition file.

   For more information, see Validating Configured Filter Definition Files on page 45.

Testing the New Business Service

This task is a step in Process of Defining Business Services on page 79. Test the new business service after you have completed all the necessary configuration and file modifications.

To test the new business service

1. Build the Sales application.

   For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

2. Start the Sales application.
3. Double-click a contact in the Contact Quick List.
4. Click the 360 icon to view the Contact Page.
5. Click the Licenses button to view the list of contact licenses.
6. Add a contact license if the list is empty.
7. Click the Delete button.

   The new Delete dialog box is displayed, prompting you to delete the contact.

8. Do one of the following:
   - Click OK and verify that the license record is deleted in the Sales application.
   - Click Cancel and verify that the license record still exists.

About Validation Rules

The Offline Client for Life Sciences provides a set of prebuilt validation rules for sample drops and signature capture in the Sales application and Message Plans in the Asset Message Planner application.

NOTE: Conversion of signatures is supported on Microsoft Windows only.

You can extend and reuse these rules, as required. The prebuilt validation rules are defined in the LSValidation.as script located, for example, in the following directory:

```
CONFIG_PROJECTS\externalservices\src\services
```

Table 4 provides a list of prebuilt validation rules available in the Offline Client for Life Sciences. All rules in this table return status and data values.

Table 4. Prebuilt Validation Rules

Validation Rule	Function	Application Used in	Related UI Operation
ActivityLicenseValidation	Validates if the activity has a valid license number.	Sales	This rule is started by the Sign button on the overlay menu on the Detailer page.
AreSamplesDroppedValid	Verifies that the quantity of samples dropped is greater than zero (0).	Sales	This rule is started by one of the following: ■ Get Signature button. ■ Save button in signature capture overlay.
IsOkToSample	Verifies if the primary contact associated with the activity can receive samples.	Sales	This rule is started by one of the following: ■ Show Sign button in overlay of Detailer Page. ■ Swap Signee button. ■ Get Signature button.
IsQtySampleValid	Verifies that the quantity of the dropped sample is less than or equal to the maximum quantity allowed for the call. It also verifies that the quantity is less than or equal to MaxQtyPerAllocation for the sales representative allocation for the product.	Sales	This rule is started by one of the following: ■ Get Signature button. ■ Save button on signature capture overlay. ■ Edit record in Sample Drop list applet.
IsStopSampling	Verifies that the Stop Sampling flag on the Sample Allocation for the product associated with the Sample Dropped Record is not set.	Sales	This rule is started by one of the following: ■ Submit Call button. ■ Get Signature button. ■ Swap Signee button.
IsValidPositionToSubmit	Verifies that the owner of the call is the current user.	Sales	This rule is started by the Submit button in the Contact Call page.

Table 4. Prebuilt Validation Rules

Validation Rule	Function	Application Used in	Related UI Operation
IsValidCallDate	Verifies that the value for the call date of the activity is today.	Sales	This rule is started by one of the following: ■ Get Signature button. ■ Show Sign in overlay menu of Detailer page. ■ Submit button in Contact Call.
StateLicenseValidation	This rule does the following: ■ Validates if the activity has a valid license number. ■ Verifies that the Contact field has a license number record for the contact address used with the current activity. The primary contact must have at least one license having the same state name as the PrimaryStateProvince field. ■ Verifies that a contact's license number exists and that the field is not null. ■ Verifies that the expiry date of a contact's license number's has not expired. ■ Verifies that the status of the contact license is Active.	Sales	This rule is started by one of the following: ■ Swap Signee button. ■ Get Signature button.

Table 4. Prebuilt Validation Rules

Validation Rule	Function	Application Used in	Related UI Operation
ValidateDiscMandatoryFlag	This rule is used when a messaging plan status is set to Approved. This rule checks if the DisclosureMandatory value is set to Y. If so, the rule then checks if the child messaging plan items and disclosure messaging plan items exist.	Asset Message Planner	This rule is started when the status is changed to Approved when editing a message plan.
ValidateReqFields	Verifies that the required fields for the contact (First Name, Last Name) are present.	Sales	This rule is started by one of the following: ■ Show Sign button in overlay menu of Detailer page. ■ Swap Signee button. ■ Submit button in Contact Call. ■ Get Signature button

Deactivating a Validation Rule

This topic describes how to deactivate a validation rule. A Validation Rule must be inactivated if it is determined not to be required to support customer's validation requirements.

To deactivate a validation rule

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click appletsdefn.xml.
3. In the appletsdefn.xml Structure window, expand AppletData, then the applets node.
4. Locate an applet where a validation rule is defined.

   For example, find and expand the APPLET - Signature Toolbar Applet node.
5. Expand the RULES node, and select the rule that you want to deactivate.
6. Change the INACTIVE attribute value to true.
7. Click Save.
8. Validate the modified definition file.

   For more information, see Validating Configured Filter Definition Files on page 45.

10 Configuring Top-Level Objects

This chapter describes how to configure top-level objects in Siebel CRM using Oracle JDeveloper. A *top-level object* refers to a parent record, such as an account or opportunity. It includes the following topics:

Roadmap for Creating a New Filter Definition File for a Top-Level Object

NOTE: This topic describes how to create a top-level, or parent, object called MedEd Event in Oracle JDeveloper. This top-level object does not exist as a predefined filter definition file in the Offline Client for Life Sciences configuration projects.

To create a new filter definition file for the MedEd Event top-level object in the Sales application, complete the following tasks as required:

Identifying Which Siebel CRM Objects to Enable for Offline Client for Life Sciences Integration

This task is a step in Roadmap for Creating a New Filter Definition File for a Top-Level Object on page 87. Complete the following procedure to identify which Siebel CRM objects to enable for integrating the Offline Client for Life Sciences. When you have identified the Siebel CRM objects to enable, you must create the integration objects accordingly. For more information, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

To identify which Siebel CRM objects to enable for Offline Client for Life Sciences integration

1. Log in to Siebel CRM as an administrator.
2. You want to enable Med Ed Events and Attendees so navigate to the MedEd screen, then the MedEd Events view.
3. Click Help, then About View.
4. In the window that displays, note down the business components for that view. For example, the business components for the Med Ed Events view are as follows:
   - Business Object: Pharma ME Event
   - Business Component: Pharma ME Event
   - Business Component: Pharma ME Event Invitee (Professional)

Obtaining WSDLs from the Server

This task is a step in Roadmap for Creating a New Filter Definition File for a Top-Level Object on page 87. Before you start creating a top-level object in Oracle JDeveloper, you must generate the MedEdEvent.wsdl file from Siebel CRM.

Generating WSDLs from Siebel CRM

Complete the following procedure to generate the MedEdEvent.wsdl file from Siebel CRM.

To generate the MedEdEvent.wsdl file from Siebel CRM

1. Log in to Siebel CRM as an administrator.
2. Navigate to the Administration - Web Services screen, then Inbound Web Services view.

3 Create a new record in the Inbound Web Services view with the Business Service Name and enter Namespace and Status values as shown, for example, in the following table.

Field	Value
Namespace	http://sdcp0000i000.corp.siebel.com
Status	Active

4 In the Service Ports applet:

a click New and create a new record with the values shown in the following table.

Field	Value
Name	Set to the actual *Business Service name*
Type	Business Service
Transport	HTTP
Binding	SOAP_DOC_LITERAL

b Create a new Business Service/Business Process Name as follows:

- Click the Look Up icon in the Business Service/Business Process Name field.
- In the popup window that displays, click New and set the values shown in the following table:

Field	Value
Name	Set to the actual *Business Service name*
Implementation Type	Business Service
Service Name/Business Process Name	IO Name

- Click OK.

5 In the Operations applet, click New and set the values shown in the following table:

Field	Value
Operation Name	Enter a unique *name* for the operation.
Method Display Name	QueryPage
Authentication	None

6 Click Clear Cache, then click Generate WSDL.

7 Download and save the WSDL file as MedEdEvent.wsdl in the WSDLFILE folder on your computer.

Creating a Filter Definition File Using the Filter Functionality

This task is a step in Roadmap for Creating a New Filter Definition File for a Top-Level Object on page 87. After you download the required WSDL files, you must create the filter definition file using the filter functionality in Oracle JDeveloper.

NOTE: For more information about filter definition files, see Key Configuration Files on page 101.

To create the filter definition file using the filter functionality in Oracle JDeveloper

1 Start Oracle JDeveloper.

2 Create a filter definition file as follows:

a Click File, New, scroll down and expand the Offline Client category, select Filter, Create New Filter, and then OK.

b Use the Create Filter File dialog that displays to enter the following information:

- **Filter File Name:** Enter a name for the filter definition file, for example, mededfilter.xml.
- **Filter Location:** Browse to the location where you want to save the filter definition file as shown, for example, in the following table.

Product	Filter Definition File Location
Siebel CRM	`CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config`

- **Time WDL File:** This file is not required for Siebel CRM.
- **WSDL File:** Browse to the location of the MedEdEvent.wsdl file.
- Click OK.

c In the following Select the DataModel dialog, do the following:

- Drag the objects that you want created for the filter from Data Objects to Selected Objects, and then click Next.
- For each Selected Object, drag the fields that you want from Available Fields to Selected Fields.
- Click Finish. Doing this generates mededfilter.xml in the specified path with the specified objects and fields.

3 Clean the filter within Oracle JDeveloper, for example, as follows:

NOTE: In this case, some additional steps must be performed in order to be able to synchronize with the Siebel CRM Server.

a Open mededfilter.xml in Oracle JDeveloper.

b Modify the following nodes:

- **Id:** Change RowId to True.
- **CreatedBy:** Change Attribute to CreatedById.
- **CreatedDate:** Change Attribute to CreatedDate.

- **Updated:** Change Attribute to ModifiedDate, and set Foreign Key to SYS_CurrentUser.
- **UpdatedBy:** Change Attribute to ModifiedById, and set Foreign Key to SYS_CurrentUser.

4 Save the file.

5 Validate the definition file.

For more information, see Validating Modified Definition Files on page 44.

Adding the Filter Definition File to a Data Source

This task is a step in Roadmap for Creating a New Filter Definition File for a Top-Level Object on page 87. After creating the filter definition file using the filter functionality in Oracle JDeveloper, you must add the filter definition file to a data source.

To add the filter definition file to a data source

1 Verify that mededfilter.xml is in the appropriate directory as shown, for example, in the following table.

Product	mededfilter.xml File Location
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config

You already created mededfilter.xml in Creating a Filter Definition File Using the Filter Functionality on page 90.

2 In the same directory, open AppCfg.as in Oracle JDeveloper and add the follow lines of code. As an example, you can search on PROMO to locate the appropriate locations for adding these lines.

a Add this line:

```
private static const APPCFG_MEDED_FILTER:String = "MEDED_FILTER_OP";
```

b Add the following two lines:

```
private static const APPCFG_MEDED_FILTER:String = "MEDED_FILTER_OP";
[Embed(source="/config/mededfilter.xml", mimeType="application/octet-
stream")] private var MEDED_FILTER:Class;
```

c Then add the following:

```
case APPCFG_MEDED_FILTER:
   bytes = new MEDED_FILTER() as ByteArray;
   xmlData = RemoveInactiveElements(bytes);
   bytes = null;
   return xmlData;
```

d Save the AppCfg.as file.

3 In the same directory, open the datasources.xml file and do the following:

a Find the data source of interest, for example Sales (OfflineClient1), and add the following line under the identified data source:

```
<SYS_EntitySyncRt id="PharmaMedEdEvent" filterLabel="MEDED_FILTER_OP"
wsdlName=" MedEdEvent.wsdl" lifetime="72" syncType="i"
fileName="mededfilter.xml">40</SYS_EntitySyncRt>
```

The *filterlabel*, which is used in the sync application, must exactly match the name given in the AppCfg.as, for example, as follows:

```
private static const APPCFG_MEDED_FILTER:String = "MEDED_FILTER_OP";.
```

The *id* comes from the WSDL entity type.

b Save the datasource.xml file.

Downloading Data with the Filter Definition File

This task is a step in Roadmap for Creating a New Filter Definition File for a Top-Level Object on page 87. After adding the filter definition file to a data source, you can download data with the filter definition file.

To download data with the filter definition file

1 Build the Sync application.

You can build the Sync application using Oracle JDeveloper or from the command line. For more information about building an application, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

2 Review the log messages to ensure that the application is ready.

3 Navigate to the folder where the CONFIG_PROJECTS are located and download and ensure that the timestamp for the Sync application is accurate.

4 Remove any existing Sync application.

Navigate to Control Panel, then select Remove Programs.

5 Remove the local database (.db) file.

For example: Sales(OfflineClient1).db.

6 Install the new Sync application.

7 Start the Sync application.

8 Select the data source that you want to connect to.

For example: Sales(OfflineClient1).

9 Click Sync and when prompted enter the appropriate login credential to connect to Sales(OfflineClient1).

10 Verify that the MedEd data has downloaded to the local database in the MedEd table.

The vCard displays the number of records.

Process of Configuring Sales for a Top-Level Object from the Filter Definition File

This task is a step in Roadmap for Creating a New Filter Definition File for a Top-Level Object on page 87. After downloading the required data with the filter definition file, you must configure the Sales application for the top-level object from the filter definition file.

To configure the Sales application for the top-level object from the filter definition file, complete the following tasks:

1. Creating a New Label on page 93
2. Updating Filter Definition File Support on page 94
3. Creating a New List Applet on page 94
4. Creating a New Page Using the New Applet on page 96
5. Adding the New Page to the Sales Application on page 97
6. Adding a Button to the Sales Application Banner on page 98

Creating a New Label

This task is a step in Process of Configuring Sales for a Top-Level Object from the Filter Definition File on page 93. To create a new label, complete the following procedure.

To create a new label

1. Start Oracle JDeveloper.
2. Open the lfs.properties file located as shown, for example, in the following table.

Product	lfs.properties File Location
Siebel CRM	CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config\enu

3. Scroll all the way to the end of the file, add the following strings, and then save the file:

```
“IDS_CUSTOM_MED_ED_EVENT" and value ="My Med Ed Events"
“IDS_CUSTOM_MED_ED_NAME" and value ="Name"
“IDS_CUSTOM_MED_ED_DESC" and value ="Description"
```

Updating Filter Definition File Support

This task is a step in Process of Configuring Sales for a Top-Level Object from the Filter Definition File on page 93. To update filter definition file support, complete the following procedure.

To update filter definition file support

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click appletsdefn.xml.
3. In the appletsdefn.xml Structure window, expand AppletData.
4. Right-click the Filters node, select Insert inside Filter, then FILTER, enter the following information in the Insert FILTER dialog that displays, then click OK.

Field	Value
NAME	Med Ed Event
VERSION	0.1

5. Save the file.
6. Validate the modified definition file.

   For more information, see Validating Configured Filter Definition Files on page 45.

Creating a New List Applet

This task is a step in Process of Configuring Sales for a Top-Level Object from the Filter Definition File on page 93. To create a new list applet, complete the following procedure.

To create a new list applet

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click appletsdefn.xml.
3. In the appletsdefn.xml Structure window, expand AppletData.
4. Create a new Med Ed Event List Applet as follows:
   a. Right-click the applets node, select Insert inside Applet, then APPLET, enter the following information in the Insert APPLET dialog that displays, and then click OK.

Field	Value
NAME	Med Ed Event List Applet
CLASS	ListAppletObj
TYPE	Standard

b Use Property Inspector to set the following attributes for APPLET - Med Ed Event List Applet:

Attribute	Value
ENTITY	MedEd
FILTER	MEDED_FILTER_OP
WIDTH	100%
HEIGHT	100%

5 Insert a template within the new applet, APPLET - Med Ed Event List Applet, as follows:

a Right-click APPLET - Med Ed Event List Applet, select Insert inside APPLET - Med Ed Event List Applet, then TEMPLATE, enter the following information in the Insert TEMPLATE dialog that displays, and then click OK.

Field	Value
WEB_TEMPLATE	A_T_Standard_List
NAME	A_T_Standard_List

b Use Property Inspector to set the following attribute values for TEMPLATE - A_T_Standard_List:

Attribute	Value
SEQUENCE	PHn

6 Add template items to the new template as follows:

a Right-click TEMPLATE - A_T_Standard_List, select Insert inside TEMPLATE - A_T_Standard_List, then TEMPLATE_ITEM.

- Enter the following information in the Insert TEMPLATE_ITEM dialog that displays, and then click OK.

Field	Value
NAME	A_T_Standard_List1
TYPE	TextInput

- Use Property Inspector to set the following attribute values for TEMPLATE_ITEM - A_T_Standard_List1:

Attribute	Value
DISPLAY_OBJECT__TEXT	Event Name
FIELD	Name

Attribute	Value
ITEM_IDENTIFIER	PHn
SEQUENCE	1

b Right click TEMPLATE - A_T_Standard_List, select Insert inside TEMPLATE - A_T_Standard_List, then TEMPLATE_ITEM. Insert another new TEMPLATE_ITEM with the following attribute values:

- Enter the following information in the Insert TEMPLATE_ITEM dialog that displays, then click OK.

Field	Value
NAME	A_T_Standard_List2
TYPE	TextInput

- Use Property Inspector to set the following attribute values for TEMPLATE_ITEM - A_T_Standard_List2:

Attribute	Value
DISPLAY_OBJECT__TEXT	Description
FIELD	EventDressCode
ITEM_IDENTIFIER	PHn
SEQUENCE	2

7 Save the file.

8 Validate the modified definition file.

For more information, see Validating Configured Filter Definition Files on page 45.

Creating a New Page Using the New Applet

This task is a step in Process of Configuring Sales for a Top-Level Object from the Filter Definition File on page 93. To create a new page using the new applet, complete the following procedure.

To create a new page using the new applet

1 Start Oracle JDeveloper.

2 In the Application Navigator window, expand the Projects option, and then click pagesdefn.xml.

3 In the pagesdefn.xml Structure window, expand the PAGES node.

4 Right-click Page - Home Page and select Copy.

5 Right-click PAGES and select Paste.

6 In the Property Inspector window, change the NAME of the new node to Med Ed Event Page.

7 Expand PAGE - Med Ed Event Page, select TEMPLATE_ITEM - Calendar Applet, and set the following attribute values:

Attribute	Value
ITEM_IDENTIFIER	Ph
APPLET	Med Ed Event Applet

8 Save the file.

9 Validate the modified definition file.

For more information, see Validating Configured Filter Definition Files on page 45.

Adding the New Page to the Sales Application

This task is a step in Process of Configuring Sales for a Top-Level Object from the Filter Definition File on page 93. To add the new page to the Sales application, complete the following procedure.

To add the new page to the Sales application

1 Start Oracle JDeveloper.

2 In the Application Navigator window, expand the Projects option, and then click applicationdefn.xml.

3 In the applicationdefn.xml Structure window, expand the APPLICATION - Sales node, then select the TEMPLATE_ITEM - Home Page node.

4 Create a new Med Ed Event List Applet as follows:

5 Right-click the TEMPLATE_ITEM - Home Page node, select Insert inside TEMPLATE_ITEM - Home Page, then PAGE.

a Enter the following information in the Insert PAGE dialog that displays:

Field Name	Value
NAME	Med Ed Event Page
THREAD_CAPTION	Med Ed Event

6 Save the file.

7 Validate the modified definition file.

For more information, see Validating Configured Filter Definition Files on page 45.

Adding a Button to the Sales Application Banner

This task is a step in Process of Configuring Sales for a Top-Level Object from the Filter Definition File on page 93. To add a button to the Sales application banner, complete the following procedure.

To add a button to the Sales application banner

1. Start Oracle JDeveloper.
2. In the Application Navigator window, expand the Projects option, and then click toolbar_buttons.xml.
3. In the toolbar_buttons.xml Structure window, expand the TOOLBARS node, then the TOOLBAR - Category_Nav_Buttons node.
4. Right-click Button - Detailer and select Copy.
5. Right-click TOOLBAR - Category_Nav_Buttons and select Paste.
6. In the Property Inspector window, set the following attribute values for the new node:

Attribute	Value
NAME	Med Ed Event
EVENTSTATE	Med Ed Event Page
WIDTH	180
ICON	flag_27x27
LABEL_TEXT	Med Ed Event
TOOLTIP_TEXT	Med Ed Event

7. Click Save.
8. Validate the modified definition file.

   For more information, see Validating Configured Filter Definition Files on page 45.

Testing the New UI for the Sales Application

When you have configured the required top-level objects, it is recommended that you test the new UI for the Sales application.

To test the new UI

1. Build the Sales application.

   For more information, see Process of Building the Offline Client for Life Sciences with Automated Scripts on page 23.

2. Start the Sales application.

3. Click the Med Ed Event button in the Sales application banner to display the new Med Ed Event list applet.

A Configuration Files, Filter Definitions, and Calculated Expressions

This appendix provides reference information for working with Oracle JDeveloper. It includes the following topics:

Key Configuration Files

Table 5 provides an overview of the configuration files in Oracle JDeveloper.

Table 5. Configuration Files in Oracle JDeveloper

File	Location	Description
Datasources.xml	Siebel CRM: `CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config`	This file does the following: ■ Defines the Siebel CRM instances (or data sources) available to the local applications. ■ Specifies the URLs for the data source. ■ Specifies the filter definition files used by an application for a specific data source. ■ Specifies the validity duration of authentication credentials. ■ Specifies the validity duration of a session before a new session ID is requested from the server. ■ Defines the synchronization period (SyncPeriod) in minutes.

Table 5. Configuration Files in Oracle JDeveloper

File	Location	Description
Filter XML files	Siebel CRM: `CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config`	This file does the following: ■ Maps Web service definitions (WSDL) to the Offline Client for Life Sciences data objects. ■ Maps Web service actions to standard the Offline Client for Life Sciences actions (for example: Insert, ChildInsert, Update, and so on). ■ Defines structural definitions for the local data store tables. The local data store table is a directory structure that contains all the local data files, including databases and preferences files. ■ Defines the general layout of the application container in the UI. ■ These are MXML-based template files containing a series of placeholders to which the application metadata definitions map.

lfs.properties	Siebel CRM:
	`CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config`
	For example, the following is the lfs.properties directory for English:
	`CONFIG_PROJECTS_OP\appcfg_sync_80x\src\config\enu`
		This file:
		■ Contains language-specific files.
		■ Stores the localized string IDs and string values used for labels and tooltips in the application.
		■ Is managed and edited through the Translation tool in Oracle JDeveloper. The English version of the lfs.properties file is available in the Offline Client for Life Sciences by default.

Application Template files	Siebel CRM: `CONFIG_PROJECTS_OP\templatesproject\page`	Defines the general layout of the application container in the UI. These are MXML-based template files containing a series of placeholders to which the application metadata definitions map.

Table 5. Configuration Files in Oracle JDeveloper

File	Location	Description
Page Template files	Siebel CRM: `CONFIG_PROJECTS_OP\templatesproject\page`	Defines the general layout of the page container in the UI. These are MXML-based template files containing a series of placeholders to which the page metadata definitions map.
Applet Template files	Siebel CRM: `CONFIG_PROJECTS_OP\templatesproject\applet`	Defines the general layout of the applet container in the UI. These are MXML-based template files containing a series of placeholders to which the applet metadata definitions map.
appletsdefn.xml	Siebel CRM: `CONFIG_PROJECTS_OP\appcfg_80x\src\config`	Defines the metadata for the applets used within an application. You use the UI builder in Oracle JDeveloper to edit and change this file.
pagesdefn.xml	Siebel CRM: `CONFIG_PROJECTS_OP\appcfg_80x\src\config`	Defines the metadata for the pages used within an application. You use the UI builder in Oracle JDeveloper to edit and change this file.
applicationdefn.xml	Siebel CRM: `CONFIG_PROJECTS_OP\appcfg_80x\src\config`	Defines the metadata for the Utilities application. You use the UI builder in Oracle JDeveloper to edit and change this file.
toolbar_buttons.xml	Siebel CRM: `CONFIG_PROJECTS_OP\appcfg_80x\src\config`	Defines the metadata for the toolbars and buttons used within an application· You use the UI builder in Oracle JDeveloper to edit and change this file.
userpreferences.xml	Siebel CRM: `CONFIG_PROJECTS_OP\appcfg_80x\src\config`	Defines the user preferences used in the Offline Client for Life Sciences. The user preferences that are not hidden are displayed in the Application's Settings dialog box.

Data Source Configuration File

This topic describes the data sources definition file in Oracle JDeveloper. For more information about the configuration files in Oracle JDeveloper, see Key Configuration Files on page 101.

Data Sources Definition File

The following topics describe how the datasources.xml file defines the specifications for all available data sources:

- Attribute Definitions of the datasources.xml File on page 104
- Schema of the datasources.xml File on page 107
- Example datasources.xml Files on page 108

Attribute Definitions of the datasources.xml File

The following tables provide a list of the attribute definitions for the datasources.xml file:

- Valid Expression String Functions, in Table 45 on page 153
- SYS_DataSources, in Table 7 on page 105
- AuthenticationValidity, in Table 8 on page 105
- SYS_EntitySyncRt, in Table 9 on page 106

Table 6 provides a list of the ListOfDataSources attributes for the datasources.xml file.

Table 6. ListOfDataSources

Attribute	Value	Function
SyncPeriod	nn	Time interval in minutes between automatic synchronization attempts if network connectivity exists.

Table 7 provides a list of the SYS_DataSources attributes for the datasources.xml file.

Table 7. SYS_DataSources

Attribute	Value	Function
Id	xxx	Identification of the data source for the user. This value is displayed in picklists, where the user can select the data source to work with.
applicationLogin	Y	Not used. Do not change.
caseSensitive	Y	Not used. Do not change.
BackupsToKeep	nn	Specifies the number of backups of the local databases that are stored.
LoginURL	xxx	Specifies the base URL to be used to communicate with the server.
LoginMethod	SiebelOP	This supports the Siebel CRM login method.
SessionLength	nn	The length of time in minutes for which a retrieved sessionId is reused, after which a new sessionId is requested.

Table 8 provides a list of the AuthenticationValidity attributes for the datasources.xml file.

Table 8. AuthenticationValidity

Attribute	Value	Function
unitOfMeasure	D	The validity of the authentication in days.
unitOfMeasure	H	The validity of the authentication in hours.
<Value>	nn	Interval in the unitOfMeasure attribute after which a user is forced to reenter authentication data.

Table 9 provides a list of the SYS_EntitySyncRt attributes for the datasources.xml file.

Table 9. SYS_EntitySyncRt

Attribute	Value	Function
Id	xxx	Interval in the unitOfMeasure after which a user is forced to re-enter authentication data.
filterlabel	xxx	Must match the corresponding filterlabel entry the in AppCfg.as file.
wsdlName	xxx	Name of the WSDL file, including the extension, associated with that specific filter definition file.
Lifetime	nn	Time interval in hours after which an object is considered stale if no successful download for that object is completed.
syncType	i	Indicates that an object is downloaded by default in incremental-download mode.
syncType	o	Indicates that an object is always downloaded in object-download mode.
Filename	The name of the filter definition XML file.	Name of the file, including the extension, holding the specifications for that specific filter definition file.
<Value>	40	The Offline Client for Life Sciences internal fixed value. Do not change.

Schema of the datasources.xml File

The following code shows the schema of a datasources.xml file. Replace the italicized items with object names, field names, or values in the datasources.xml file:

```
ListOfDataSources: <ListOfDataSources SyncPeriod="nn">
  <SYS_DataSource>
     SYS_DataSource
  </SYS_DataSource>

SYS_DataSource: <SYS_DataSource id="xxx"
                applicationLogin="Y|N"
                caseSensitive="y|N"
                BackupsToKeep="nn">
  <LoginURL>xxx</LoginURL>
  <LoginMethod>SiebelOD</LoginMethod>
  <SessionLength>nn</SessionLength>
  <MessageType>xxx<</MessageType>
  // For Siebel CRM, xxx is Flat.
  <AuthenticationValidity>AuthenticationValidity</AuthenticationValidity>
  <ListOfEntitySyncRts>ListOfEntitySyncRts</ListOfEntitySyncRts>
</SYS_DataSource>

AuthenticationValidity: <AuthenticationValidity unitOfMeasure="D">nn
</AuthenticationValidity>

ListOfEntitySyncRts: <ListOfEntitySyncRts>
  SYS_EntitySyncRt
</ListOfEntitySyncRts>

SYS_EntitySyncRt: <SYS_EntitySyncRt id="xxx" filterLabel="xxx"
  wsdlName="xxx"
  lifetime="nn"
  syncType="x"
  fileName="xxx">
  40
</SYS_EntitySyncRt>
```

Example datasources.xml Files

This topic shows the structure of the datasources.xml file for Siebel CRM. The characters xxx in the code represent where variables are used.

Example datasources.xml for Siebel CRM

The following is an example datasources.xml file for Siebel CRM.

```
<ListOfDataSources SyncPeriod="60">

<SYS_DataSource id="Sales (CRM On Premise)" applicationLogin="Y" caseSensitive="n"
BackupsToKeep="1" timeout="">

   <LoginURL>http://xx.xxx.xxx.xxx</LoginURL>

   <LoginMethod>SiebelOP</LoginMethod>

   <SessionLength>10</SessionLength>

   <MessageType>Flat</MessageType>

   <AuthenticationValidity unitOfMeasure="D">5</AuthenticationValidity>

   <ListOfEntitySyncRts>

      <SYS_EntitySyncRt id="LSSGOP_Account" filterLabel="ACCOUNT_FILTER_OP"
      wsdlName="http___siebel.com_asi__LSSGOP_Account.WSDL" lifetime="72" syncType="i"
      fileName="accountfilter.xml">40</SYS_EntitySyncRt>

      <SYS_EntitySyncRt id="LSSGOP_AccountAddress"
      filterLabel="ACCOUNTADDRESS_FILTER_OP"
      wsdlName="http___siebel.com_asi__LSSGOP_AccountAddress.WSDL" lifetime="72"
      syncType="i" fileName="accountaddressfilter.xml">40</SYS_EntitySyncRt>

      <SYS_EntitySyncRt id="LSSGOP_Activity" filterLabel="ACTIVITY_FILTER_OP"
      wsdlName="http___siebel.com_asi__LSSGOP_Activity.WSDL" lifetime="6" syncType="o"
      fileName="activityfilter.xml">40</SYS_EntitySyncRt>

      <SYS_EntitySyncRt id="LSSGOP_ActivityContact"
      filterLabel="ACTIVITYCONTACT_FILTER_OP"
      wsdlName="http___siebel.com_asi__LSSGOP_ActivityContact.WSDL" lifetime="6"
      syncType="o" fileName="activitycontactfilter.xml">40</SYS_EntitySyncRt>

      <SYS_EntitySyncRt id="LSSGOP_PersonalList" filterLabel="ALLOCATION_FILTER_OP"
      wsdlName="http___siebel.com_asi__LSSGOP_PersonalList.WSDL" lifetime="72"
      syncType="i" fileName="allocationfilter.xml">40</SYS_EntitySyncRt>

      <SYS_EntitySyncRt id="LSSGOP_Contact" filterLabel="CONTACT_FILTER_OP"
      wsdlName="http___siebel.com_asi__LSSGOP_Contact.WSDL" lifetime="72" syncType="i"
      fileName="contactfilter.xml">40</SYS_EntitySyncRt>

      <SYS_EntitySyncRt id="LSSGOP_ContactAddress"
      filterLabel="CONTACTADDRESS_FILTER_OP"
      wsdlName="http___siebel.com_asi__LSSGOP_ContactAddress.WSDL" lifetime="72"
      syncType="i" fileName="contactaddressfilter.xml">40</SYS_EntitySyncRt>

      <SYS_EntitySyncRt id="LSSGOP_ContactAccount"
      filterLabel="CONTACTACCOUNT_FILTER_OP"
      wsdlName="http___siebel.com_asi__LSSGOP_ContactAccount.WSDL" lifetime="72"
      syncType="i" fileName="contactaccountfilter.xml">40</SYS_EntitySyncRt>
```

```
<SYS_EntitySyncRt id="LSSGOP_ContactBestTimes"
filterLabel="CONTACTBESTTIMES_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_ContactBestTimes.WSDL" lifetime="72"
syncType="i" fileName="contactbesttimesfilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_ContactLicense"
filterLabel="CONTACTLICENSE_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_ContactLicense.WSDL" lifetime="72"
syncType="i" fileName="contactlicensefilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_Literature" filterLabel="SOLUTION_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_Literature.WSDL" lifetime="9"
syncType="i" fileName="solutionfilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_MessagePlan" filterLabel="MESSAGE_PLAN_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_MessagePlan.WSDL" lifetime="9"
syncType="i" fileName="messageplanfilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_MessageResponse"
filterLabel="MESSAGERESPONSE_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_MessageResponse.WSDL" lifetime="6"
syncType="o" fileName="messageresponsefilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_MsgPlanItem" filterLabel="MSGPLANITEM_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_MsgPlanItem.WSDL" lifetime="9"
syncType="i" fileName="messageplanitemfilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_MsgPlanItemRelation"
filterLabel="MSGPLANITEMRELATION_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_MsgPlanItemRelation.WSDL" lifetime="9"
syncType="i" fileName="msgplanitemrelationfilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_Product" filterLabel="PRODUCT_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_Product.WSDL" lifetime="72" syncType="i"
fileName="productfilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_ProductIndication"
filterLabel="PRODUCT_INDICATION_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_ProductIndication.WSDL" lifetime="72"
syncType="i" fileName="productindicationfilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_ProductsDetailed"
filterLabel="PRODUCTSDETAILED_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_ProductsDetailed.WSDL" lifetime="6"
syncType="o" fileName="productsdetailedfilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_PromotionalItemsDropped"
filterLabel="PROMOTIONALITEMSDROPPED_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_PromotionalItemsDropped.WSDL"
lifetime="6" syncType="o" fileName="promotionalitemsdroppedfilter.xml">40</
SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_RelatedAccount"
filterLabel="RELATEDACCOUNT_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_RelatedAccount.WSDL" lifetime="72"
syncType="i" fileName="relatedaccountfilter.xml">40</SYS_EntitySyncRt>

<SYS_EntitySyncRt id="LSSGOP_RelatedContact"
filterLabel="RELATEDCONTACT_FILTER_OP"
wsdlName="http___siebel.com_asi__LSSGOP_RelatedContact.WSDL" lifetime="72"
syncType="i" fileName="relatedcontactfilter.xml">40</SYS_EntitySyncRt>
```

```
        <SYS_EntitySyncRt id="LSSGOP_RelatedLiterature"
        filterLabel="RELATEDSOLUTION_FILTER_OP"
        wsdlName="http___siebel.com_asi__LSSGOP_RelatedLiterature.WSDL" lifetime="9"
        syncType="i" fileName="relatedsolutionfilter.xml">40</SYS_EntitySyncRt>

        <SYS_EntitySyncRt id="LSSGOP_RetailObjective"
        filterLabel="RETAILOBJECTIVE_FILTER_OP"
        wsdlName="http___siebel.com_asi__LSSGOP_RetailObjective.WSDL" lifetime="6"
        syncType="o" fileName="retailobjectivefilter.xml">40</SYS_EntitySyncRt>

        <SYS_EntitySyncRt id="LSSGOP_SampleDisclaimer"
        filterLabel="SAMPLE_DISCLAIMER_FILTER_OP"
        wsdlName="http___siebel.com_asi__LSSGOP_SampleDisclaimer.WSDL" lifetime="48"
        syncType="i" fileName="sampledisclaimerfilter.xml">40</SYS_EntitySyncRt>

        <SYS_EntitySyncRt id="LSSGOP_SampleDropped"
        filterLabel="SAMPLEDROPPED_FILTER_OP"
        wsdlName="http___siebel.com_asi__LSSGOP_SampleDropped.WSDL" lifetime="6"
        syncType="o" fileName="sampledroppedfilter.xml">40</SYS_EntitySyncRt>

        <SYS_EntitySyncRt id="LSSGOP_SampleInventory"
        filterLabel="SAMPLE_INVENTORY_FILTER_OP"
        wsdlName="http___siebel.com_asi__LSSGOP_SampleInventory.WSDL" lifetime="72"
        syncType="i" fileName="sampleinventoryfilter.xml">40</SYS_EntitySyncRt>

        <SYS_EntitySyncRt id="LSSGOP_SampleTransaction"
        filterLabel="SAMPLE_TRANSACTION_FILTER_OP"
        wsdlName="http___siebel.com_asi__LSSGOP_SampleTransaction.WSDL" lifetime="72"
        syncType="i" fileName="sampletransactionfilter.xml">40</SYS_EntitySyncRt>

        <SYS_EntitySyncRt id="LSSGOP_Signature" filterLabel="SIGNATURE_FILTER_OP"
        wsdlName="http___siebel.com_asi__LSSGOP_Signature.WSDL" lifetime="6"
        syncType="o" fileName="signaturefilter.xml">40</SYS_EntitySyncRt>

        <SYS_EntitySyncRt id="LSSGOP_TransactionItem"
        filterLabel="TRANSACTIONITEM_FILTER_OP"
        wsdlName="http___siebel.com_asi__LSSGOP_TransactionItem.WSDL" lifetime="72"
        syncType="i" fileName="transactionitemfilter.xml">40</SYS_EntitySyncRt>

    </ListOfEntitySyncRts>

</SYS_DataSource>

<SYS_DataSource id="AMP (CRM On Premise)" applicationLogin="Y" caseSensitive="n"
BackupsToKeep="1" timeout="">

    <LoginURL>http://xx.xxx.xxx.xxx</LoginURL>

    <LoginMethod>SiebelOP</LoginMethod>

    <SessionLength>10</SessionLength>

    <MessageType>Flat</MessageType>

    <AuthenticationValidity unitOfMeasure="D">5</AuthenticationValidity>

    <ListOfEntitySyncRts>

        <SYS_EntitySyncRt id="LSSGOP_CatalogCategory" filterLabel="BOOK_FILTER_OP"
        wsdlName="http___siebel.com_asi__LSSGOP_CatalogCategory.WSDL" lifetime="9"
        syncType="i" fileName="bookfilter.xml">40</SYS_EntitySyncRt>
```

```
    <SYS_EntitySyncRt id="LSSGOP_MessagePlan"
    filterLabel="MESSAGE_PLAN_FILTER_OP_AMP"
    wsdlName="http___siebel.com_asi__LSSGOP_MessagePlan.WSDL" lifetime="9"
    syncType="i" fileName="messageplanfilter_AMP.xml">40</SYS_EntitySyncRt>

    <SYS_EntitySyncRt id="LSSGOP_MessagePlanCatalog"
    filterLabel="MESSAGE_PLAN_BOOK_FILTER_OP"
    wsdlName="http___siebel.com_asi__LSSGOP_MessagePlanCatalog.WSDL" lifetime="9"
    syncType="i" fileName="messageplanbookfilter.xml">40</SYS_EntitySyncRt>

    <SYS_EntitySyncRt id="LSSGOP_MsgPlanItem"
    filterLabel="MSGPLANITEM_FILTER_OP_AMP"
    wsdlName="http___siebel.com_asi__LSSGOP_MsgPlanItem.WSDL" lifetime="9"
    syncType="i" fileName="messageplanitemfilter_AMP.xml">40</SYS_EntitySyncRt>

    <SYS_EntitySyncRt id="LSSGOP_MsgPlanItemRelation"
    filterLabel="MSGPLANITEMRELATION_FILTER_OP_AMP"
    wsdlName="http___siebel.com_asi__LSSGOP_MsgPlanItemRelation.WSDL" lifetime="9"
    syncType="i" fileName="msgplanitemrelationfilter_AMP.xml">40</SYS_EntitySyncRt>

    <SYS_EntitySyncRt id="LSSGOP_Product" filterLabel="PRODUCT_FILTER_OP"
    wsdlName="http___siebel.com_asi__LSSGOP_Product.WSDL" lifetime="72" syncType="i"
    fileName="productfilter.xml">40</SYS_EntitySyncRt>

    <SYS_EntitySyncRt id="LSSGOP_RelatedLiterature"
    filterLabel="RELATEDSOLUTION_FILTER_OP"
    wsdlName="http___siebel.com_asi__LSSGOP_RelatedLiterature.WSDL" lifetime="9"
    syncType="i" fileName="relatedsolutionfilter.xml">40</SYS_EntitySyncRt>

    <SYS_EntitySyncRt id="LSSGOP_Literature" filterLabel="SOLUTION_FILTER_OP_AMP"
    wsdlName="http___siebel.com_asi__LSSGOP_Literature.WSDL" lifetime="9"
    syncType="i" fileName="solutionfilter_AMP.xml">40</SYS_EntitySyncRt>

    <SYS_EntitySyncRt id="LSSGOP_LiteratureCatalog"
    filterLabel="SOLUTIONBOOK_FILTER_OP"
    wsdlName="http___siebel.com_asi__LSSGOP_LiteratureCatalog.WSDL" lifetime="9"
    syncType="i" fileName="solutionbookfilter.xml">40</SYS_EntitySyncRt>

   </ListOfEntitySyncRts>

</SYS_DataSource>

</ListOfDataSources>
```

Filter Definitions

This topic provides information about filter definitions. It includes the following:

For more information about filter definition files for Siebel CRM, see *Siebel CRM Web Services Reference*.

About Filter Definition Files

Filter definition files are configuration files. They are used for the following purposes:

- Defining the data being exchanged with the server
- Defining the hierarchy of that data
- Defining certain characteristics of data elements

For more information on filter definition files, see Chapter 5, "Getting Started with Oracle JDeveloper."

Filter Definition Files Available with the Offline Client for Life Sciences

This topic lists some of the filter definition files available with Siebel CRM and shows some of the important attributes.

Filter Definition Files Available with Siebel CRM

See Table 10 for the filter definition files available with Siebel CRM and some of the filter attributes. See Table 11 on page 114 and Table 12 on page 115 for additional attributes of the filter definition files.

Table 10. Siebel CRM Filter Definition Files

Filter Definition	File Name	Configured Search Specification?	Application
Account	accountaddressfilter.xml	No	Sales
Account	accountfilter.xml	No	Sales
Activity	activitycontactfilter.xml	No	Sales
Activity	activityfilter.xml	Yes	Sales

Table 10. Siebel CRM Filter Definition Files

Filter Definition	File Name	Configured Search Specification?	Application
Address	addressfilter.xml	No	Sales
Allocation	allocationfilter.xml	Yes	Sales
Book	bookfilter.xml	No	Asset Message Planner
Contact Account	conactaccountfilter.xml	No	Sales
Contact Address	contactaddressfilter.xml	No	Sales
Contact Best Times	contactbesttimesfilter.xml	No	Sales
Contact	contactfilter.xml	No	Sales
Contact License	contactlicensefilter.xml	No	Sales
Message Plan Book	messageplanbookfilter.xml	No	Asset Message Planner
Message Plan	messageplanfilter.xml	Yes	Sales
Message Plan (AMP)	messageplanfilter_AMP.xml	No	Asset Message Planner
Message Plan Item	messageplanitemfilter.xml	Yes	Sales, Asset Message Planner
Message Plan Item Relation	msgplanitemrelationfilter.xml	Yes	Sales
Message Response	messageresponsefilter.xml	Yes	Sales
Product	productfilter.xml	No	Sales, Asset Message Planner
Production Indication	productindicationfilter.xml	No	Sales
Products Detailed	productsdetailedfilter.xml	Yes	Sales
Promotional Items Dropped	pomotionalitemsdroppedfilter.xml	Yes	Sales
PSR Account	psraccountfilter.xml	No	Sales
PSR Contact	psrcontactfilter.xml	No	Sales
Related Account	relatedaccountfilter.xml	No	Sales
Related Contact	relatedcontactfilter.xml	No	Sales
Related Solution	relatedsolutionfilter.xml	No	Sales
Retail Objective	retailobjectivefilter.xml	No	Sales

Table 10. Siebel CRM Filter Definition Files

Filter Definition	File Name	Configured Search Specification?	Application
Revenue	revenuefilter.xml	No	Sales
Sample Disclaimer	sampledisclaimerfilter.xml	Yes	Sales
Sample Dropped	sampledroppedfilter.xml	Yes	Sales
Sample Inventory	Sampleinventoryfilter.xml	Yes	Sales
Sample Transaction	sampletransactionfilter.xml	Yes	Sales
Signature	signaturefilter.xml	Yes	Sales
Solution Book	solutionbookfilter.xml	No	Sales
Solution	solutionfilter.xml	Yes	Sales
Solution (AMP)	solutionfilter_AMP.xml	No	Asset Message Planner
Transaction Item	transactionitemfilter.xml	Yes	Sales

Table 11 describes additional attributes of the filter definitions available with Siebel CRM.

Table 11. Siebel CRM Filter Definition Files - Additional Attributes

Filter Definition	Data Source Entry	Web Service
Account	LSSGOP_Account	Account
Account	LSSGOP_AccountAddress	AccountAddress
Activity	LSSGOP_Activity	Activity
Activity	LSSGOP_ActivityContact	ActivityContact
Address	LSSGOP_Address	Address
Allocation	LSSGOP_PromotionalItemsDropped	Allocation
Allocation	LSSGOP_SampleDropped	Allocation
Book	LSSGOP_CatalogCategory	Book
Contact	LSSGOP_Contact	Contact
Contact Account	LSSGOP_ContactAccount	ContactAccount
Contact Address	LSSGOP_ContactAddress	ContactAddress
Contact Best Times	LSSGOP_ContactBestTimes	ContactBestTimes
Contact License	LSSGOP_ContactLicense	ContactLicense
Message Plan	LSSGOP_MessagePlan	MessagePlan
Message Plan (AMP)	LSSGOP_MessagePlan	MessagePlan

Table 11. Siebel CRM Filter Definition Files - Additional Attributes

Filter Definition	Data Source Entry	Web Service
Message Plan Item	LSSGOP_MsgPlanItem	MsgPlanItem
Message Plan Item Relation	LSSGOP_MsgPlanItemRelation	MsgPlanItemRelation
Message Response	LSSGOP_MessageResponse	MessageResponse
Product	LSSGOP_Product	Product
Product Indication	LSSGOP_ProductIndication	ProductIndication
Products Detailed	LSSGOP_ProductsDetailed	ProductsDetailed
Promotional Items Dropped	LSSGOP_PromotionalItemsDropped	PromotionalItemsDropped
Related Account	LSSGOP_RelatedAccount	RelatedAccount
Related Contact	LSSGOP_RelatedContact	RelatedContact
Related Solution	LSSGOP_RelatedLiterature	RelatedLiterature
Retail Objective	LSSGOP_RetailObjective	RetailObjective
Revenue	LSSGOP_Revenue	Revenue
Sample Disclaimer	LSSGOP_SampleDisclaimer	SampleDisclaimer
Sample Dropped	LSSGOP_SampleDropped	SampleDropped
Sample Inventory	LSSGOP_SampleInventory	SampleInventory
Sample Transaction	LSSGOP_SampleTransaction	SampleTransaction
Signature	LSSGOP_Signature	Signature
Solution	LSSGOP_Literature	Solution
Solution book	LSSGOP_LiteratureCatalog	SolutionCatalog
Solution (AMP)	LSSGOP_Literature	Solution
Transaction Item	LSSGOP_TransactionItem	TransactionItem

Table 12 shows the search specifications of the filter definitions available with the Sales application for Siebel CRM.

Table 12. Siebel CRM Filter Definitions - Search Specifications

Filter Definition	Search Specification
Activity	([DueDate]>=ADDDAY(TODAY(),-180) AND [DueDate]<=ADDDAY(TODAY(),15))
Allocation (for Promotional Items Dropped and Sample Dropped)	([OwnerId]=CURRENTUSERID() AND [DueDate]>=ADDDAY(TODAY(),-180) AND [DueDate]<=ADDDAY(TODAY(),15))"

Table 12. Siebel CRM Filter Definitions - Search Specifications

Filter Definition	Search Specification
Message Plan	([Status]="Released" AND ([EndDate]>TODAY() OR [EndDate]=""))
Message Plan Item	([ParentMessagePlanStatus]="Released" AND ([EndDate]>TODAY() OR [EndDate]=""))
Message Plan Item Relation	([ParentMessagePlanStatus]="Released" AND ([EndDate]>TODAY() OR [EndDate]=""))">
Message Response	([OwnerId]=CURRENTUSERID() AND [DueDate]>=ADDDAY(TODAY(),-180) AND [DueDate]<=ADDDAY(TODAY(),15))
Products Detailed	([OwnerId]=CURRENTUSERID() AND [DueDate]>=ADDDAY(TODAY(),-180) AND [DueDate]<=ADDDAY(TODAY(),15))
Promotional Items Dropped	([OwnerId]=CURRENTUSERID() AND [DueDate]>=ADDDAY(TODAY(),-180) AND [DueDate]<=ADDDAY(TODAY(),15))
Sample Disclaimer	([Status]="Active")
Sample Dropped	([OwnerId]=CURRENTUSERID() AND [DueDate]>=ADDDAY(TODAY(),-180) AND [DueDate]<=ADDDAY(TODAY(),15))
Sample Inventory	([ActiveFlag]=TRUE())
Sample Transaction	([OwnerId]=CURRENTUSERID() AND [DueDate]>=ADDDAY(TODAY(),-180) AND [DueDate]<=ADDDAY(TODAY(),15))>
Signature	([OwnerId]=CURRENTUSERID() AND [DueDate]>=ADDDAY(TODAY(),-180) AND [DueDate]<=ADDDAY(TODAY(),15))
Solution	([DistributionMethod]="N" AND [ReleaseDate]<TIMESTAMP() AND ([ExpirationDate]>TODAY() OR [ExpirationDate]=""))
Transaction Item	([OwnerId]=CURRENTUSERID() AND [DueDate]>=ADDDAY(TODAY(),-180) AND [DueDate]<=ADDDAY(TODAY(),15))

Filter Attribute Definitions

This topic describes filter attribute definitions:

ObjectFilter Attributes

Table 13 describes ObjectFilter attributes for the Filter schema.

Table 13. ObjectFilter Attributes

Attribute Name	Required or Optional	Default Value	Value
WSID	Required	None	OP
WSID	Required	None	OP

TopLevelObjectList Attributes

Table 14 describes TopLevelObjectList attributes for the Filter schema.

Table 14. TopLevelObjectList Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
Inactive	Optional	N	Y	The data in this attribute is not part of the data model for the client application.
Inactive	Optional	N	N	The data in this attribute is part of the data model for the client application.
NoSync	Optional	N	Y	The data in this attribute is for use in the local computer only. It will not be downloaded from or uploaded to the Siebel CRM Server.

Table 14. TopLevelObjectList Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
NoSync	Optional	N	N	The data in this attribute is downloaded from the Siebel CRM Server, and local updates are uploaded to the server.
Proprietoryship	Optional	Client	Client	Initiates updates in the local computer to override the updates in the server in the event of conflict.
Proprietoryship	Optional	Client	Server	Prevents updates in the local computer overriding the updates in the server in the event of conflict.
Type	Required	None	EntityList	The value EntityList is a list of objects.

TopLevelObject Attributes

Table 15 describes TopLevelObject attributes for the Filter schema.

Table 15. TopLevelObject Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
CascadeDelete	Optional	N	Y	Deletes all child records when the parent record is deleted.
CascadeDelete	Optional	N	N	Prevents the deletion of child records when the parent record is deleted.
Entity	Required	None	*Object*	Specifies the name of the object; that is, the name of the local database table.
Inactive	Optional	N	Y	The data in this attribute is not part of the data model for the client application.
Inactive	Optional	N	N	The data in this attribute is part of the data model for the client application.
NoSync	Optional	N	Y	The data in this attribute is for use in the local computer only. It is not downloaded from or uploaded to the Siebel CRM Server.

Table 15. TopLevelObject Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
NoSync	Optional	N	N	The data in this attribute is downloaded from the server, and local updates are uploaded to the Siebel CRM Server.
SearchSpec	Optional	N	*Expression*	Downloads only records meeting the condition specified in the value Expression from the Siebel CRM Server. If this attribute is not specified, all records accessible by the user are downloaded.
Type	Required	None	Entity	This value Entity is an object.

ObjectList Attributes

Table 16 describes 1MObjectList attributes for the Filter schema.

Table 16. 1MObjectList Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
Associate	Optional	N	Y	Associates records with or disassociates records from a parent object.
Associate	Optional	N	N	Disables the association of a record with or disassociation of a record from a parent object.
CascadeDelete	Optional	N	Y	Deletes all child records when the parent record is deleted.
CascadeDelete	Optional	N	N	Disables the deletion of child records when the parent record is deleted.
Inactive	Optional	N	Y	The data in this attribute is not part of the data model for the client application.
Inactive	Optional	N	N	Indicates that the data is part of the data model for the client application.
Insert	Optional	Y	Y	Creates new records for this object. Records are removed from the local database after deletion.

Table 16. 1MObjectList Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
Insert	Optional	Y	N	Disables the creation of new records for this object.
NoSync	Optional	N	Y	Indicates that data is for use in the local computer only. Synchronization does not occur. Data is not downloaded from or uploaded to the server.
NoSync	Optional	N	N	Indicates that data is downloaded from the server and local updates are uploaded to the server. Synchronization is allowed.
Type	Required	None	EntityList	The value EntityList is a list of objects.

ObjectList Attributes

Table 17 describes MMObjectList attributes for the Filter schema.

Table 17. MMObjectList Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
Associate	Optional	N	Y	Associates records with or to disassociate records from a parent object.
Associate	Optional	N	N	Prevents the association of records with or the disassociation of records from a parent object.
CascadeDelete	Optional	N	Y	Deletes all child records when the parent record is deleted.
CascadeDelete	Optional	N	N	Deletes no child records when the parent record is deleted.
ChildKey	Required	None	*Object*Id1, *Object*Id2	The *Object*Id1 and *Object*Id2 values are foreign keys in the intersection table. These values correspond to the list in ParentKey attribute.

Table 17. MMObjectList Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
Entity	Required	None	*Object*	Indicates the name of the intersection table of the many-to-many relationship.
Inactive	Optional	N	Y	Indicates that the data is not part of the data model for the client application.
Inactive	Optional	N	N	Indicates that the data is part of the data model for the client application.
Insert	Optional	Y	Y	Indicates that new records can be created for this object. Records are removed from the local database when they are deleted.
Insert	Optional	Y	N	Indicates that new records cannot be created for this object.
NoSync	Optional	N	Y	Indicates that data is for use in the local computer only. Data is not downloaded from or uploaded to the server. No synchronization occurs.
NoSync	Optional	N	N	Indicates that data is downloaded from the server and local updates are uploaded to the server. Synchronization does occur.
ParentKey	Required	None	*Object*1.*Object*Id1, *Object*2.*Object*Id2	The values in the ParentKey attribute are *joined-in* fields, used to form many-to-many relationships.
Type	Required	None	EntityList	This value EntityList is a list of objects.

Object Attributes

Table 18 describes object attributes for the Filter schema.

Table 18. Object Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
CascadeDelete	Optional	N	Y	Deletes all child records when the parent record is deleted.
CascadeDelete	Optional	N	N	Deletes no child records when the parent record is deleted.
ChildKey	Required	None	ObjectId	The ChildKey attribute is the foreign key for joining in the parent object.
Entity	Required	None	Object	Specifies the name of the object; that is, the name of the local database table.
ParentKey	Required	None	Object.ObjectId	The values in the ParentKey attribute are *joined-in* fields for the parent object.
Type	Required	None	Entity	The value Entity is an object.

Field Attributes

Table 19 describes Field attributes for the Filter schema.

Table 19. Field Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
Attribute	Required	None	*Field*	Specifies the name of the field in the local database.
AttributeDestination	Optional	None	*Field*	Specifies the target object field to be joined in.
CascadeDelete	Optional	N	Y	Deletes all child records when the parent record is deleted.
CascadeDelete	Optional	N	N	Deletes no child records when the parent record is deleted.

Table 19. Field Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
Data	Optional	N	Y	This is a joined-in field of binary data type.
Data	Optional	N	N	This is not a joined-in field of binary data type.
Datatype	Optional	None	DTYPE_BOOL	Specifies that the field data type is Boolean.
Datatype	Optional	None	DTYPE_BLOB	Specifies that the field data type is Binary Large Object (BLOB) data for images.
Datatype	Optional	None	DTYPE_CURRENCY	Specifies that the field data type is Currency.
Datatype	Optional	None	DTYPE_DATE	Specifies that the field data type is Date only.
Datatype	Optional	None	DTYPE_DATETIME	Specifies that the field data type is Date and time.
Datatype	Optional	None	DTYPE_ID	Specifies that the field data type is Identifier field.
Datatype	Optional	None	DTYPE_INTEGER	Specifies that the field data type is Integer.
Datatype	Optional	None	DTYPE_NOTE	Specifies that the field data type is Text.
Datatype	Optional	None	DTYPE_NUMBER	Specifies that the field data type is Float.
Datatype	Optional	None	DTYPE_PHONE	Specifies that the field data type is Phone number.
Datatype	Optional	None	DTYPE_TEXT	Specifies that the field data type is Text.
Datatype	Optional	None	DTYPE_TIME	Specifies that the field datatype is Time only.
Datatype	Optional	None	DTYPE_REAL	Real number.
Datatype	Optional	None	DTYPE_UTCDATETIME	Date and time in Coordinated Universal Time (UTC) format.
Datatype	Optional	None	DTYPE_XML	XML
Datalength	Optional	None	*Number*	Specifies the length of field data in bytes.

Table 19. Field Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
DefaultValue	Optional	None	*Value*	Specifies a default value for the field.
DestinationKey	Optional	None	*Field*	The field located in the joined-in object, which is used to identify the record to be joined in.
Entity	Required	None	*Object*	Specifies the name of the object; that is, the name of the local database table.
Filterable	Optional	None	Y	This attribute is applicable to WSDL. The value *Y* means that this field is searchable.
Filterable	Optional	None	N	This attribute is applicable to WSDL . The value *N* means that this field is not searchable.
ForeignKeyTo	Optional	None	*Field*	Links the Field value to a record in another object. Can be used for establishing a one-to-many relationship.
Inactive	Optional	N	Y	The data in this attribute is not part of the data model for the client application.
Inactive	Optional	N	N	The data in this attribute is part of the data model for the client application.
LOVType	Optional	None	*LOVType*	Specifies the LOV type (List of Value type) for a LOV picklist.
NoColumn	Optional	N	Y	Indicates that no database column exists for this field.
NoColumn	Optional	N	N	Indicates that a database column does exist for this field.

Table 19. Field Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
NoSync	Optional	N	Y	The data in this attribute is for use in the local computer only. It is not downloaded from or uploaded to the server.
NoSync	Optional	N	N	The data in this attribute is downloaded from the server and local updates are uploaded to the server.
ParentLOVField	Optional	None	*ParentLOVField*	The parent LOV language-independent code for hierarchical picklists.
ReadOnly	Optional	N	Y	This field is read-only. Set this attribute to Y for joined-in fields.
ReadOnly	Optional	N	N	This field is updatable.
Required	Optional	N	Y	Indicates that the field is a required field and cannot be left blank.
Required	Optional	N	N	This field is not required.
RowId	Required	N	Y	Specifies that an Id field is required for the object. There must be only one Id field for each object. The Id field usually has the name *Id* in the WSDL.
RowId	Required	N	N	Specifies that an Id field is not required for the object.
SearchSpec	Optional	N	*Expression*	Specifies that only records matching the value *Expression* are downloaded from the server. If this attribute is not specified, all records accessible by the user are downloaded.
SourceKey	Optional	None	*Field*	Specifies the foreign key needed to perform a join-in between a join-in field and another object.

Table 19. Field Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
SystemField	Optional	N	Y	This field is generated by the Offline Client for Life Sciences.
SystemField	Optional	N	N	This field is not generated by the Offline Client for Life Sciences.
Type	Required	None	*Field*	This element is a field.
UserKey	Optional	Null	0	This field is the user key. You can use only one user key for each object, which is usually the Id field.
UserKey	Optional	Null	Null	This field is not the user key.

Field Constraint

Table 20 describes the field constraint for the Filter schema.

Table 20. Field Constraint

Attribute Name	Required or Optional	Default Value	Value	Function
ID	Required	None	UNIQUE	There can be only one record with this value in this field.
ID	Required	None	NOT NULL	Indicates that this field must not be empty.

Table Constraints for the Filter Schema

Table 21 describes the table constraints for the Filter schema.

Table 21. Table Constraint

Attribute Name	Required or Optional	Default Value	Value	Function
Entity	Required	None	Object	Name of the object this contraint is applied to
UniqueConstraint	Required	None	*Field*[, *Field*]*	Specifies one or more fields which make up the unique constraint for this object.

WebService and ServiceAPI Attribute Definitions

The WebService and ServiceAPI elements map WSDL function names. A WebService element maps the data access function names defined in the data object WSDL field. A ServiceAPI element maps the Siebel CRM Service functions names defined in the Service API WSDL.

This topic describes the WebService and Service API attribute definitions:

Function Attributes

Table 22 describes the attributes used for Function.

Table 22. Function Attributes

Attribute Name	Required or Optional	Default Value	Value	Function (and Web Services Version Supported)
Function	Required	None	*Function*[,*Function*]*	The following functions are currently defined for the attribute Function: ■ QueryPage ■ Execute (2.0) ■ Insert ■ InsertChild (1.0) ■ Update ■ UpdateChild (1.0) ■ Delete ■ DeleteChild (1.0)
Name	Required	None	ObjectFunction_Input	This is the required attribute for the <Input> tag. For more information, see the attribute name Function.
Name	Required	None	ObjectFunction _Output	This is the required attribute for the <Output> tag. For more information, see the attribute name Function.

Element Attributes

Table 23 list the values for the attribute called Name.

Table 23. Element Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
Name	Required	None	ListOfObject	List of objects for which the function is defined.
Name	Required	None	LOVLanguageMode	Specifies the language mode for picklists. The language mode can be language-independent or language-dependent.
Name	Required	None	ViewMode	Specifies the level of access to records specified in the method call.
Name	Required	None	Echo	Controls whether data sent to Siebel CRM through integration Web services are recorded as transactions.

Application, Page, and Applet Definitions

The user interface metadata for Offline Client for Life Sciences applications is defined in a series of XML-based definition files. You use Oracle JDeveloper to edit and to display the content of these XML-based definition files. This topic specifies the schema and the attributes of each tag contained in each of the XML-based reference files. It includes the following information:

Application Definition

The following code provides an example of the application definition.

Example of Code for Application Schema Definition

The following code is an example of an application schema definition:

```
applicationdefn.xml: <APPLICATION>
                       <TEMPLATE_ITEM>+
                     </APPLICATION>
<TEMPLATE_ITEM>:     <TEMPLATE_ITEM>
                       <PAGE>*
                     </TEMPLATE_ITEM>
```

The following topics describe the attributes for the application schema:

<Application> Attributes

Table 24 describes the <Application> attributes for the application schema definition.

Table 24. <Application> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
HELP_APPLET	Optional	None	Text	Name of the applet that displays online help text.
LOCK_APPLET	Optional	None	Text	Name of the applet that locks the screen.

Table 24. <Application> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
NAME	Required	None	Text	Application name.
TEMPLATE	Required	None	Text	Template for the application.

<TEMPLATE_ITEM> Attributes

Table 25 describes the <TEMPLATE_ITEM> attributes for the application schema definition.

Table 25. <TEMPLATE_ITEM> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
ITEM_IDENTIFIER	Required	None	*Text*	Placeholder name in the template where this item is displayed.
NAME	Required	None	*Text*	Name of the item.
NAME_TEXT	Optional	None	*Text*	Displayed text for this item.

Type	Optional	None	`Signature`
			`Slider`
			`SortControl`
			`Text`
			`Text10`
			`Text11`
			`Text12`
			`Text13`
			`Text14`
			`Text8`
			`Text9`
			`TextArea`
				Each value specifies a UI component in the Offline Client for Life Sciences.

Type (contd)	Optional	None	`TextBold`
			`TextBold10`
			`TextBold11`
			`TextBold12`
			`TextBold13`
			`TextBold14`
			`TextBold8`
			`TextBold9`
			`TextInput`
			`ThreadBar`
			`ToggleButtonBar`
			`Toolbar`
			`TrashControl`
			`UnboundedTextArea`
			`UnboundedTextInput`
			`VerticalButtonBar`
			`VerticalButtonBarLeft`
			`Vtoolbar`
				Each value specifies a UI component in the Offline Client for Life Sciences.

Table 25. <TEMPLATE_ITEM> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
TYPE	Optional	None	AdvancedCheckBox AdvancedColorPicker AdvancedCombobox AdvancedTextArea AppletToggle CheckBox ColorPicker ComboBox CustomCtrl DateField DatePickerField DateTimeField Detailer DrillDownThreadBar HorizontalButtonBar HorizontalNumericStepper HToolbar IconMap Image Label Label10 Label11 Label12 Label13 Label14 Label8 Label9 LabelBold LabelBold10 LabelBold11 LabelBold12 LabelBold13 LabelBold14 LabelBold8 LabelBold9 MultilineText MultilineText2 MultiSelectCheckBox MultiSelectComboBox Number NumericStepper Page PDQControl PDQControlPanelPick PopUpButton PwdControl	Each value specifies a UI component in the Offline Client for Life Sciences.
TYPE (contd)	Optional	None	RadioButton Ratings	Each value specifies a UI component in the Offline Client for Life Sciences.

<PAGE> Attributes

Table 26 describes the <PAGE> attributes for the application schema definition.

Table 26. <PAGE> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
CAPTION	Optional	None	*Text*	Caption for the applet in a page.
GROUP	Optional	None	*Text*	Name of the toggle applet group. Applets with the same group are under the same AppletToggle control.
NAME	Required	None	*Text*	Name of the page.
SEQUENCE	Optional	None	*Number*	Specifies the order of the applet in the ToggleApplet control.
THREADVISIBLE	Optional	None	True	Indicates this thread is visible. In this context, thread refers to a list of previously visited pages.
THREADVISIBLE	Optional	None	False	Indicates this thread is not visible.
THREAD_CAPTION	Optional	None	*Text*	Not used.
THREAD_CAPTION_TEXT	Optional	None	*Text*	Not used.

Page Definition

The following code example describes the general definition of a page schema.

Example Code for Page Schema Definition

The following code example describes the structure of a page schema definition:

```
aagesdefn.xml:          <PAGES>
                          <PAGE>+
                        </PAGES>

<PAGE>:                 <PAGE>
                          <TEMPLATE>
                        </Page>

<TEMPLATE>:             <TEMPLATE>
                          <SECTION>*
                          <TEMPLATE_ITEM>+
                         </TEMPLATE>

<TEMPLATE_ITEM>:        <TEMPLATE_ITEM>
                          <USERPROPS>?
                          <TOGGLEAPPLETS>?
                          <TOGGLEPARTNERS>?
                         </TEMPLATE_ITEM>

<USERPROPS>:            <USERPROPS>
                          <USERPROP>*
                         </USERPROPS>

<TOGGLEAPPLETS>:        <TOGGLEAPPLETS>
                          <TOGGLEAPPLET>*
                         </TOGGLEAPPLETS>

<TOGGLEAPPLET>:         <TOGGLEAPPLET>
                          <TOGGLEPARTNERS>?
                         </TOGGLEAPPLET>

<TOGGLEPARTNERS>:       <TOGGLEPARTNERS>
                          <TOGGLEPARTNER>*
                         </TOGGLEPARTNERS>
```

The following topics describe the attributes for the page schema definition:

- <PAGE> Attributes on page 134
- <TEMPLATE> Attributes on page 134
- <SECTION> Attribute on page 134
- <TEMPLATE_ITEM> Attributes on page 135
- <USERPROP> Attributes on page 136
- <TOGGLEAPPLET> Attributes on page 137

■ <TOGGLEPARTNER> Attributes on page 138

<PAGE> Attributes

Table 27 describes the <PAGE> attributes for the page schema definition.

Table 27. <PAGE> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
MODE	Required	None	ModalPopUp	This page is a pop-up window relating to the MODE attribute.
MODE	Required	None	PopUp	This page is a pop-up window, which does not relate to the MODE attribute.
MODE	Required	None	Standard	This is a standard page.
NAME	Required	None	*Text*	Name of the page.

<TEMPLATE> Attributes

Table 28 describes the <TEMPLATE> attribute for the page schema definition.

Table 28. <TEMPLATE> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
NAME	Required	None	*Text*	Name of a page template.

<SECTION> Attribute

Table 29 describes the <SECTION> attribute for the PAGE definition.

Table 29. <SECTION> Attribute

Attribute Name	Required or Optional	Default Value	Value	Function Description
NAME	Required	None	*Text*	Name of the section.

<TEMPLATE_ITEM> Attributes

Table 30 describes the <TEMPLATE_ITEM> attributes for the page schema definition.

Table 30. <TEMPLATE_ITEM> Attribute

Attribute Name	Required or Optional	Default Value	Value	Function Description
APPLET	Required	None	*Text*	Name of the applet to be included in this page.
CAPTION	Optional	None	*Text*	String ID of the applet title.
CAPTION_TEXT	Optional	None	*Text*	Text of the applet title.
DEEPCOPY	Optional	None	N	Cannot use a deep copy. **NOTE:** *Deep copy* refers to copying an object including its child objects.
DEEPCOPY	Optional	None	Y	Can use a deep copy.
DISPLAY_COUNT	Optional	None	Number	Cached page size.
EQUIVALENTTO	Optional	None	*Text*	Name of another applet to which this applet is equivalent.
EQUIVALENT_APPLET_FLD	Optional	None	*Text*	The field to establish the equivalency. *Equivalency* is a tightly coupled one-to-one relationship between two applets of the same object type. It is used to simulate several levels in a hierarchy. For example, you have a Contact object with an activity Child Object. You also have an activity Object with a SampleDropped Child object. To simulate the hierarchy Contact - Activity - SampleDropped you can set up equivalency between the activity child object and the activity Top Level object. This links the two Activity objects tightly; that is, modifications made to one will automatically reflect in the other.

Table 30. <TEMPLATE_ITEM> Attribute

Attribute Name	Required or Optional	Default Value	Value	Function Description
GROUP	Optional	None	*Text*	Identifies which Toggle button group to which this applet belongs.
ICON	Optional	None	*Text*	Name of the icon to be displayed in the Toggle button for this applet.
NAME	Required	None	*Text*	Indicates a placeholder in the page template where this applet is displayed.
PARENT_APPLET	Optional	None	*Text*	Name of the parent applet.
SEQUENCE	Optional	None	Number	Sequence of the applet within its toggle applet group.
THIS_APPLET_FLD	Optional	None	*Text*	Used for Equivalent Applets to match the parent and child equivalent applets.
TOOLTIP	Optional	None	*Text*	String Id for the tooltip for the Toggle button.
TOOLTIP_TEXT	Optional	None	Text	Tooltip text for the Toggle button.
VISIBLE	Optional	Y	N	This applet is not visible.
			Y	This applet is visible.

<USERPROP> Attributes

Table 31 describes the <USERPROP> attributes for the page schema definition.

Table 31. <USERPROP Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
NAME	Required	None	ToggleAppletsGroup	Defines a toggle applet group.
VALUE	Required	None	*Text*	Name of toggle applet group.

<TOGGLEAPPLET> Attributes

Table 32 describes the <TOGGLEAPPLET> attributes for the page schema definition.

Table 32. <TOGGLEAPPLET> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
APPLET	Required	None	*Text*	Name of the applet to be included in this page.
CAPTION	Optional	None	*Text*	String ID of the applet title.
CAPTION_TEXT	Optional	None	*Text*	Text of the applet title.
DISPLAY_COUNT	Optional	None	*Number*	Cached page size.
EQUIVALENTTO	Optional	None	*Text*	Name of another applet that this applet is equivalent to.
EQUIVALENT_APPLET_FLD	Optional	None	*Text*	The field to establish the equivalency. Equivalency is a tightly coupled one-to-one relationship between two applets of the same object type. It is used to simulate n-level hierarchies. For example, you have a Contact object with an activity Child Object. You also have an activity Object with a SampleDropped Child object. To simulate the hierarchy Contact - Activity - SampleDropped you can set up equivalency between the activity child object and the activity Top Level object. This links the two Activity objects very tightly, that is, modifications made to one will automatically reflect also in the other.
GROUP	Optional	None	*Text*	Identifies which Toggle button group this applet belongs to.
ICON	Optional	None	*Text*	Name of the icon to be displayed in the Toggle button for this applet.
PARENT_APPLET	Optional	None	*Text*	Name of the parent applet.
SEQUENCE	Optional	None	*Number*	Sequence of the applet within its toggle applet group.

Table 32. <TOGGLEAPPLET> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
THIS_APPLET_FLD	Optional	None	*Text*	Used for Equivalent Applets to match the parent and child equivalent applets.
TOOLTIP	Optional	None	*Text*	String Id for the tooltip for the Toggle button.
TOOLTIP_TEXT	Optional	None	*Text*	Tooltip text for the Toggle button.
VISIBLE	Optional	Y	N	This applet is not visible.
VISIBLE	Optional	Y	Y	This applet is visible.

<TOGGLEPARTNER> Attributes

Table 33 describes the <TOGGLEPARTNER> attributes for the page schema definition.

Table 33. <TOGGLEPARTNER> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
APPLET	Required	None	*Text*	Name of the applet serving as the toggle partner.
ITEM_IDENTIFIER	Required	None	*Text*	Placeholder in template where this applet is displayed.
VISIBLE	Optional	Y	N	This applet is not visible.
VISIBLE	Optional	Y	Y	This applet is visible.

Applets Definition

The following code sample describes a general definition of an applet schema.

Example of Code for Applets Schema Definition

The following code is an example of an applets schema definition.

```
appletsdefn.xml:      < AppletData>
                        < Filters>
                        < applets>
                      </ AppletData>

< Filters>:           <Filters>
                        <FILTER>+
                      </Filters>

<applets>:           <applets>
                        <APPLET>+
                     </applets>

<APPLET>:             <APPLET>
                        <DRILLDOWN_OBJECTS>?
                        <RULES>?
                        <APPLET_USER_PROPS>?
                        <TEMPLATE>
                      </APPLET>

<DRILLDOWN_OBJECTS>: <DRILLDOWN_OBJECTS>
                        <DRILLDOWN_OBJECT>*
                      </DRILLDOWN_OBJECTS>

<DRILLDOWN_OBJECT>:  <DRILLDOWN_OBJECT>
                        <DYNAMIC_DRILLDOWN_DESTIONATIONS>*
                      </DRILLDOWN_OBJECT>

<DYNAMIC_DRILLDOWN_DESTINATIONS>: <DYNAMIC_DRILLDOWN_DESTINATIONS>
                                    DYNAMIC_DRILLDOWN_DESTINATION>*
                                  </DYNAMIC_DRILLDOWN_DESTINATIONS>

<RULES>:              <RULES>
                        <RULE>*
                      </RULES>

<APPLET_USER_PROPS>: <APPLET_USER_PROPS>
                        <APLLET_USER_PROP>*
                      </APPLET_USER_PROPS>

<TEMPLATE>:           <TEMPLATE>
                        <TEMPLATE_ITEM>+
                      </TEMPLATE>

<TEMPLATE_ITEM>:      <TEMPLATE_ITEM>
                        <TEMPLATE_ITEMS_USER_PROPS>*
                        <PICKMAPS>*
                      </TEMPLATE_ITEM>

<TEMPLATE_ITEMS_USER_PROPS>: <TEMPLATE_ITEMS_USER_PROPS>
                               <TEMPLATE_ITEMS_USER_PROP>*
                             </TEMPLATE_ITEMS_USER_PROPS>

<PICKMAPS>:           <PICKMAPS>
                        <MAPPING>*
                      </PICKMAPS>
```

The following topics describe the attributes for the page schema definition:

<FILTER> Attributes

Table 34 describes the <FILTER> attributes for the Applets schema definition.

Table 34. <FILTER> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
NAME	Required	None	*Text*	Name of the filter definition file.
VERSION	Required	None	*Number*	Version number of the filter definition file.

<applets> Attribute

Table 35 describes the <applets> attribute for the Applets schema definition.

Table 35. <applets> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
SEQUENCE	Optional	None	*Number*	Not used.

<APPLET> Attributes

Table 36 describes the <APPLET> attributes for the Applets schema definition.

Table 36. <APPLET> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
ALPHA_FIELD	Optional	None	*Text*	Field for filtering records using the alphabetical query bar.
ASSOCIATE_APPLET	Optional	None	*Text*	Not used.
CLASS	Required	None	AppletObj	This applet is a generic applet.
CLASS	Required	None	FormAppletObj	This applet is a form.
CLASS	Required	None	ListAppletObj	This applet is a list.
CLASS	Required	None	PickAppletObj	This applet is for selection through query and filtering.
CLASS	Required	None	PivotObj	This applet is for pivot query. A *pivot query* is a query that delivers the sum total of a number of values.
COMMENTS	Optional	None	*Text*	Information about the applet.
DRAG_ENABLED	Optional	None	*Text*	Indicates that a user can move records in this applet another applet.
DROP_ENABLED	Optional	None	*Text*	This applet can be a drop target for relevant records. This refers to a drag and drop action, which requires a source to drag from and a target to drop the object to.
ENTITY	Required	None	*Text*	The object or database table that this applet is based on.
FILTER	Required	None	*Text*	The filter definition file on which this applet is based.
HEIGHT	Optional	0	*Number*	Height in pixels.
HEIGHT	Optional	0	*Number%*	Height expressed as a percentage.

Table 36. <APPLET> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
HELP_IDENTIFIER	Optional	None	*Text*	Unique identifier in an index of help files.
INACTIVE	Optional	N	N	This applet is active.
INACTIVE	Optional	N	Y	This applet is disabled.
INSERT_POSITION	Optional	First	first	Inserts new records at the beginning of the list.
INSERT_POSITION	Optional	First	last	Inserts new records at the end of the list.
INSERT_POSITION	Optional	First	immediateAfter	Inserts new records immediately after the selected record.
NAME	Required	None	*Text*	Name of the applet.
NO_DELETE	Optional	N	N	Records can be deleted.
NO_DELETE	Optional	N	Y	Records cannot be deleted.
NO_INSERT	Optional	N	N	Records can be inserted.
NO_INSERT	Optional	N	Y	Records cannot be inserted.
NO_UPDATE	Optional	N	N	Records can be modified.
NO_UPDATE	Optional	N	Y	Records cannot be modified.
PERSISTENT	Optional	N	N	The content and layout information of the applet are not cached.
PERSISTENT	Optional	N	Y	The content and layout information of the applet are cached for the session. The cache prevents the applet from being reloaded each time. Use this attribute for data or UI intensive applets. *UI intensive applets* are applets with a complicated UI, requiring lengthy processes to reload the applet.
SEARCH_SPEC	Optional	None	*Text*	Only display records that meet the filtering criteria.
TITLE	Optional	None	*Text*	String Id for the applet title.

Table 36. <APPLET> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
TITLE_TEXT	Optional	None	*Text*	Displayed text of the applet title.
TYPE	Optional	Standard	Associate	Not used.
TYPE	Optional	Standard	Standard	Specifies that this is a normal applets.
TYPE	Optional	Standard	Pick	Specifies that this type is a pick map.
WIDTH	Optional	0	*Number*	Width in pixels.
WIDTH	Optional	0	*Number%*	Width expressed as a percentage.
XMLDATA_ROOT_TAG	Required	None	ListOfObject	Name of the XML tag name in the filter definition file for the entity.

<DRILLDOWN_OBJECT> Attributes

Table 37 describes the <DRILLDOWN_OBJECT> attributes for the Applets schema definition.

Table 37. <DRILLDOWN_OBJECT> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
COMMENTS	Optional	None	*Text*	Indicates a comment field.
DESTINATION_FIELD	Required	None	*Text*	Field in the target record that matches the field of the source record.
ENTITY	Required	None	*Text*	Entity of the target record.
HYPERLINK_FIELD	Required	None	*Text*	Field in this record for initiating the drilldown.
INACTIVE	Optional	N	N	This drilldown is active.
INACTIVE	Optional	None	Y	This drilldown is disabled.
NAME	Required	None	*Text*	Name of the drilldown
PAGE	Required	None	*Text*	Page to drilldown to, or the link you click to obtain more detailed information.

Table 37. <DRILLDOWN_OBJECT> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
SEQUENCE	Optional	None	*Number*	This is the order of evaluation among the list of drilldowns. Each drilldown in the list is evaluated in order of the SEQUENCE attribute. If a drilldown is found, where all conditions are fulfilled, that drilldown is executed and further evaluation of the list is stopped.
SOURCE_FIELD	Required	None	*Text*	Field in this applet for matching the destination field.

<DYNAMIC_DRILLDOWN_DESTINATION> Attributes

Table 38 describes the <DYNAMIC_DRILLDOWN_DESTINATION> attributes for the Applets schema definition.

Table 38. <DYNAMIC_DRILLDOWN_DESTINATION> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Descriptions
DESTINATION_DRILLDOWN_OBJECT	Required	None	*Text*	Name of drilldown object to be evaluated next.
FIELD	Required	None	*Text*	Name of the field from which the drilldown is based.
INACTIVE	Optional	N	N	Dynamic drilldown destination is active.
INACTIVE	Optional	N	Y	Dynamic drilldown destination is disabled.
NAME	Required	None	*Text*	Name of the dynamic drilldown destination.
SEQUENCE	Optional	None	*Number*	This is the order of evaluation among the list of drilldowns. Each drilldown in the list is evaluated in order of the SEQUENCE attribute. If a drilldown is found, where all conditions are fulfilled, that drilldown is executed and further evaluation of the list is stopped.
VALUE	Required	None	*Text*	Value of the field on which the drilldown is based.

<RULE> Attributes

Table 39 describes the <RULE> attributes for the Applets schema definition.

Table 39. <RULE> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
CLASS	Required	None	*Text*	This is the ActionScript class where the rule is defined. For a rule to perform any action in the Offline Client for Life Sciences application, it must be associated with corresponding ActionScript code. This code is contained in a file, representing a class. Each class can contain multiple code segments, or methods, that perform different actions.
ENTITY	Optional	None	*Text*	The entity to which the rule is applied.
EVENT	Required	None	*Text*	The UI event that triggers the rule.
INACTIVE	Optional	N	N	The rule is active.
INACTIVE	Optional	N	Y	The rule is disabled.
METHOD	Required	None	*Text*	This is the ActionScript class where the rule is defined. For a rule to perform any action in the Offline Client for Life Sciences application, it must be associated with corresponding ActionScript code. This code is contained in a file, representing a class. Each class can contain multiple code segments, or methods, that perform different actions.
NAME	Required	None	*Text*	Name of the rule.

<APPLET_USER_PROP> Attributes

Table 40 describes the <APPLET_USER_PROP> attributes for the Applets schema definition.

Table 40. <APPLET_USER_PROP> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
INACTIVE	Optional	N	N	This applet user property is active.
INACTIVE	Optional	N	Y	This applet user property is disabled.
NAME	Required	None	*Text*	Name of the applet user property.
VALUE	Required	None	*Text*	Value of the applet user property.

<TEMPLATE> Attributes

Table 41 describes the <TEMPLATE> attributes for the Applets schema definition.

Table 41. <TEMPLATE> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
INACTIVE	Optional	N	N	This applet user property is active.
INACTIVE	Optional	N	Y	This applet user property is disabled.
NAME	Required	None	*Text*	Id for this template.
SEQUENCE	Required	None	*Text*	Placeholder that references the main control of any template for loading of subcontrols, for example, in the form applet template the placeholder, PHn, refers to the tile container into which all form items are loaded. A *tile container* is a placeholder for a section on the template, into which all form items are loaded.
SUBTMPL	Optional	None	*Text*	Name of the subtemplate, which is used for a pop up.
WEB_TEMPLATE	Required	None	*Text*	Name of the applet template used for this applet.

<TEMPLATE_ITEM> Attributes

Table 42 describes the <TEMPLATE> attributes for the Applets schema definition.

Table 42. <TEMPLATE_ITEM> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
AVAILABLE	Optional	None	N	This item is in the default display. Applicable to List Applets only.
AVAILABLE	Optional	None	Y	The item is in the Available list of the applet rather than the default display. This attribute is applicable to list applets only.
CALCULATED	Optional	N	N	This item maps to a field in the entity.
CALCULATED	Optional	N	Y	This is a calculated field and the formula is in EXPRESSION.
COLINDEX	Optional	None	*Number*	Indicates the form column number in which this item is displayed. Column numbering starts from 0.
COLSPAN	Optional	None	*Number*	Number of columns this item occupies.
COMMENTS	Optional	None	*Text*	Information about the item.
DATA_TYPE	Required	None	DTYPE_BOOL	Specifies a Boolean data type.
DATA_TYPE	Required	None	DTYPE_CURRENCY	Specifies a Currency data type.
DATA_TYPE	Required	None	DTYPE_DATE	Specifies a Date only data type.
DATA_TYPE	Required	None	DTYPE_DATETIME	Specifies a Date and time data type.
DATA_TYPE	Required	None	DTYPE_ID	Specifies a ID data type.

Table 42. <TEMPLATE_ITEM> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
DATA_TYPE	Required	None	DTYPE_INTEGER	Specifies a Integer data type.
DATA_TYPE	Required	None	DTYPE_NUMBER	Specifies a Float data type.
DATA_TYPE	Required	None	DTYPE_PHONE	Specifies a Phone number data type.
DATA_TYPE	Required	None	DTYPE_TEXT	Specifies a Text data type.
DATA_TYPE	Required	None	DTYPE_TIME	Specifies a Time only data type.
DATA_TYPE	Required	None	DTYPE_ZIPCODE	Specifies a ZIP code data type.
DISPLAY_FORMAT	Optional	None	*Text*	Display format for the item.
DISPLAY_OBJECT	Optional	None	*Text*	Id of an additional displayed item (icon or string).
DISPLAY_OBJECT_TEXT	Optional	None	*Text*	Text (title or label).
EXPRESSION	Required when CALCULATED is set to Y.	None	*Text*	Formula to calculate the value of the item.
FIELD	Optional	None	*Text*	Field in the entity for this item.
GROUP	Optional	None	*Text*	This item belongs to a specific group, such as a popup.
HEIGHT	Optional	0	*Number*	Height in pixels.
INACTIVE	Optional	N	N	This item is active.
INACTIVE	Optional	N	Y	This item is disabled.
ITEM_IDENTIFIER	Optional	None	*Text*	Specifies in which placeholder this item is displayed in the applet template.
NAME	Required	None	*Text*	Name of the item.

Table 42. <TEMPLATE_ITEM> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
NO_COPY	Optional	None	N	This field can be copied. If DEEPCOPY is set to Y on the child applet, then the child records will also be copied.
NO_COPY	Optional	None	Y	This field cannot be copied.
PICK_APPLET	Optional	None	*Text*	Applet that provides fields to define the pick map.
POPUP_EDIT	Optional	N	N	This item displays in a standard format.
POPUP_EDIT	Optional	N	Y	This item is displayed in a bigger popup for ease of editing.
PRE_DEFAULT	Optional	None	*Text*	Default value for this item after creation of a new record.
READ_ONLY	Optional	N	N	The item is editable.
READ_ONLY	Optional		Y	The item is read-only.
REQUIRED	Optional	N	N	The item is not required.
REQUIRED	Optional	N	Y	The item is required.
ROWINDEX	Optional	None	*Number*	Row number of the item in the form. It starts at zero (0).
ROWSPAN	Optional	None	*Number*	Number of columns this item occupies.
SEQUENCE	Optional	None	*Number*	The order of this item in the applet.
SHOW_IN_LIST	Optional	None	N	Indicates that this item cannot be displayed, for example, a Delete button in the list. Applicable to list applet only.

Table 42. <TEMPLATE_ITEM> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description
SHOW_IN_LIST	Optional	None	Y	Displays this item in the list. Applicable to list applet only.
SORT	Optional	None	*Number*	Sorts items in a consecutive numerical order, starting from 1.
SORT_MODE	Optional	None	ASC	Sorts in ascending order.
SORT_MODE	Optional	None	DESC	Sorts in descending order.
TAB_STOP	Optional	None	None	Not used
TEXT_ALIGNMENT	Optional	left	center	Aligns text in the center.
TEXT_ALIGNMENT	Optional	left	left	Left aligned.
TEXT_ALIGNMENT	Optional	left	right	Right aligned.

TYPE	Optional	None	`AdvancedCheckBox`
			`AdvancedColorPicker`
			`AdvancedCombobox`
			`AdvancedTextArea`
			`AppletToggle`
			`CheckBox`
			`ColorPicker`
			`ComboBox`
			`CustomCtrl`
			`DateField`
			`DatePickerField`
			`DateTimeField`
			`Detailer`
			`DrillDownThreadBar`
			`HorizontalButtonBar`
			`HorizontalNumericStepper`
			`HToolbar`
			`IconMap`
			`Image`
			`Label`
			`Label10`
			`Label11`
			`Label12`
			`Label13`
			`Label14`
			`Label8`
			`Label9`
			`LabelBold`
			`LabelBold10`
			`LabelBold11`
			`LabelBold12`
			`LabelBold13`
			`LabelBold14`
			`LabelBold8`
			`LabelBold9`
				UI control for displaying this item.

Table 42. <TEMPLATE_ITEM> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function Description

TYPE (contd)	Optional	None	`MultilineText`
			`MultilineText2`
			`MultiSelectCheckBox`
			`MultiSelectComboBox`
			`Number`
			`NumericStepper`
			`Page`
			`PDQControl`
			`PDQControlPanel`
			`Pick`
			`PopUpButton`
			`PwdControl`
			`RadioButton`
			`Ratings`
			`Signature`
			`Slider`
			`SortControl`
			`Text`
			`Text10`
			`Text11`
			`Text12`
			`Text13`
			`Text14`
			`Text8`
			`Text9`
			`TextArea`
			`TextBold`
			`TextBold10`
			`TextBold11`
			`TextBold12`
			`TextBold13`
			`TextBold14`
			`TextBold8`
			`TextBold9`
			`TextInput`
			`ThreadBar`
			`ToggleButtonBar`
			`Toolbar`
			`TrashControl`
			`UnboundedTextArea`
			`UnboundedTextInput`
			`VerticalButtonBar`
			`VerticalButtonBarLeft`
			`Vtoolbar`
				UI control for displaying this item.

VISIBLE	Optional	None	N	Displays this item.
VISIBLE	Optional	None	Y	Does not display this item.
WIDTH	Optional	0	*Number*	Width of this item in pixels.

<TEMPLATE_ITEMS_USER_PROP> Attributes

Table 43 describes the <TEMPLATE_ITEMS_USER_PROP> attributes for the Applets schema definition.

Table 43. <TEMPLATE_ITEMS_USER_PROP> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
INACTIVE	Optional	N	N	This user property is active.
INACTIVE	Optional	N	Y	This user property is disabled.
NAME	Required	None	*Text*	Name of the template item user property.
VALUE	Required	None	*Text*	Value of the template item user property.

<MAPPING> Attributes

Table 44 describes the <MAPPING> attributes for the Applets schema definition.

Table 44. <MAPPING> Attributes

Attribute Name	Required or Optional	Default Value	Value	Function
CONSTRAINT	Optional	N	N	There is no field constraint on the Pick Applet query.
CONSTRAINT	Optional	N	Y	The Pick Applet query is constrained by records where the FIELD in the pick applet matches the value in the PICK_FIELD.
FIELD	Required	None	*Text*	The field in this applet to be mapped.
INACTIVE	Optional	N	N	The mapping attribute is active.
INACTIVE	Optional	N	Y	The mapping attribute is disabled.
PICK_FIELD	Required	None	*Text*	The mapped field in the pick applet.

Calculated Expressions

You can define calculated fields in applet definitions using the UI builder functionality and the calculated expressions supported by Oracle's Offline Client for Life Sciences. An *expression* can be a literal expression, a function, or a field. Currently, there is no limit to the number of functions that can be nested. It is recommended that you consider the performance or memory limits when building an expression. This topic provides some sample expressions.

While function names are case insensitive, use a mixture of uppercase and lowercase for readability, for example, *GetPreferenceChoice*. Nonetheless, the function names are converted to uppercase.

Numeric values can be either enclosed in double quotation marks or not. A minus sign (-) must be beside the numeric value.

Predefined Functions

Expression Parser supports a limited number of predefined functions.

The following tables describe the functions that are identified as valid expression functions used for calling the Expression Parser:

- Valid Expression String Functions on page 153
- Valid Expression XML Functions on page 154
- Valid Expression Date Functions on page 154
- Valid Expression Math Functions on page 156
- Valid Expression System Functions on page 158

Table 45 describes valid expression string functions.

Table 45. Valid Expression String Functions

String Function Name	Return Value	Description
Concat(expr1, expr2)	String	Returns the concatenation of expr1 and expr2.
GetPreference("identifier")	String	Gets the value of the preference associated with the identifier.
Language()	String	Specifies the preferred language value.
Left(expr, integer)	String	Leftmost integer of the expression. If the integer is empty, the default is 1.
LowerCase(expr)	String	Converts the expression to lowercase.
LPad(expr1,integer,expr2)	String	Adds the <expr2> repeatedly to the left of <expr1> until the result reaches the length of <integer> characters.
Replace(expr1,expr2,expr3)	String	Replaces each occurrence of expr2 in expr1 with expr3.

Table 45. Valid Expression String Functions

String Function Name	Return Value	Description
Right(expr, integer)	String	Rightmost integer of the expression.
SubString(expr, integer1, integer2)	String	Returns the substring portion of the expression starting in position of integer1, returning the length of integer2.
Trim(expr)	String	Deletes all leading and trailing spaces from the expression.
UpperCase(expr)	String	Converts the expression to uppercase.

Table 46 describes valid expression XML functions.

Table 46. Valid Expression XML Functions

XML Function Name	Return Value	Description
Count([expr])	Number	Counts the number of child elements in the data structure passed in with the argument expr. If no argument expr is specified, the function counts the number of child elements in the data structure. You must set this prior to calling Count() using the function SetXMLData(xmlData).
ParentField("<fieldname>")	String	The field name enclosed in quotation marks. For example, ParentField("Location") will return the value of the field Location on the parent xml node. If no value is returned, then the result is an empty string value.

Table 47 describes valid expression date functions.

Table 47. Valid Expression Date Functions

Date Function Name	Return Value	Description
AddHrs(expr, integer)	Date	Adds the number of integer hours to the expression expr that must evaluate to a date or time and returns the result. For example, AddHrs(TimeStamp(),1) returns the time one hour from the current time.

Table 47. Valid Expression Date Functions

Date Function Name	Return Value	Description
AddDay(expr, integer)	Date	Adds the number of integer days to the expression expr that must evaluate to a date or time and returns the result. For example, AddDay(TimeStamp(),1) returns tomorrow's date. The value of the expression must resolve to a date.
AddMins(expr, integer)	Date	Adds the number of integer minutes to the expression expr that must evaluate to a date or time and returns the result. For example, AddMins(TimeStamp(),1) returns the time it will be in one minute from now.
AddSecs(expr, integer)	Date	Adds the number of integer seconds to the expression expr that must evaluate to a date or time and returns the result. For example, AddSecs(TimeStamp(),1) returns the time it will be in one second from the current time.
AddWeek(expr, integer)	Date	Adds the number of integer weeks to the expression expr that must evaluate to a date or time and returns the result. For example, AddWeek(TimeStamp(),1) returns the date it will be in one week from today.
AddYear(expr, integer)	Date	Adds the number of integer years to the expression expr that must evaluate to a date or time and returns the result. For example, AddYear(TimeStamp(),1) returns the date it will be in one year from today.
DateSequence(expr1,expr2)	Boolean	Returns true if either of the values for expr1 or expr2 is null, or, if the value of expr1 is less than, or equal to, expr2. The values for expr1 and expr2 must be date fields.
Day(expr)	Int	Returns the day of the month for the expression. The value of expr must evaluate to a date.
DayOfWeek(expr)	Int	Returns the numeric equivalent of the day of the week based on system values of the expression. The expression must evaluate to a date.
DaysBetween(expr1, expr2)	Int	Returns the number of days between expr1 and expr2. The values for expr1 and expr2 must evaluate to a date.

Table 47. Valid Expression Date Functions

Date Function Name	Return Value	Description
Month(expr)	Int	Returns the numeric equivalent of the month in the year. This value is calculated by adding a value of one to the system value. Alternatively, this is done by the calculating the date range for the month by taking the first day of the month from the last of the month for the expression. When a date range is used, it is usually a query-based request.
SubTime(expr1,expr2)	String	Subtracts the values expr2 from expr1. The two values for expr1 and expr2 must be in the DateTime format. This value returns the time difference in milliseconds, which is converted to string format.
TimeStamp()	Date	Full timestamp of the system date. TimeStamp() returns a date variable representing the current date and time (including milliseconds).
Today()	Date	System date value formatted.
Year(expr)	Int	Returns the fullYear attribute of the expression. The value of the expression must evaluate to a date.

Table 48 describes valid expression math functions.

Table 48. Valid Expression Math Functions

Math Function Name	Return Value	Description
Count([expr])	Number	Counts the number of children in the data passed in the expression. Alternatively, if no data is passed, this function counts the XML data set before calling the expression parser.
ParentField("<fieldname>")	String	Indicates the field name enclosed in quotation marks. For example, ParentField("Location") returns the value of the field location on the parent XML node. If no value is returned, then the result is an empty string value.

Table 48. Valid Expression Math Functions

Math Function Name	Return Value	Description
IIf(condition expr, true expr, false expr)	Object	The condition expression must evaluate to a Boolean. If the condition evaluates to True, the result of the true expression is returned. Otherwise, the result of the false expression is returned. The default value of the condition expression is false.
IfNull(expr1, expr2)	Type of expr1	Returns the value of expr1 unless expr1 is NULL, in which case the value of expr2 is returned.
LookUpName(LOV Type, Display Value, [Parent LIC])	String	Returns the Language Independent Code (LIC) of Display Value based on LOV Type, (Parent LIC is optional). ■ Indicates the value of LOV Type. The type code for the LOV is usually the concatenation of entity name (for example, Account) and picklist field name (for example, AccountType) as follows: Account\AccountType. ■ Indicates the display value, which is the language dependent display value of the value you want to convert. ■ Indicates the parent Language Independent Code (LIC) of the parent pick field (only for hierarchical picklists).
LookUpValue(LOV Type, LIC, [Parent LIC])	String	Returns the Display Value for a given LIC: ■ Indicates the LOV Type. The type code for the LOV is usually the concatenation of entity name (for example, Account) and picklist field name (for example, AccountType) as follows: Account\AccountType. ■ Indicates the LIC of the value to convert. ■ Indicates the parent LIC. This is the LIC of the parent pick field (only for hierarchical picklists) This returns the characters "" for non hierarchical picklists.

Table 49 describes valid expression system functions.

Table 49. Valid Expression System Functions

System Function Name	Return Value	Description
Current_User	String	Returns the current user ID value.
GetSessionAttribute(expr)	String	Gets the session attribute value of expr.
GetPreferences(expr)	String	Gets the preference value associated with expr.
Not(expr)	Boolean	Returns a value of True or False based on the value of expr, which must evaluate to a Boolean.
True()	String	Returns True as a value.
False()	String	Returns False as a value.

About the Expression Parser Class

The expression parser class has the following public functions in addition to the *constructor*:

NOTE: In object-oriented programming, a *constructor* is a type of subroutine called at the creation of an object that prepares the object for use.

- **ExpressionParser.** Accepts the expression string as input.
- **CreateTopLevel.** Accepts the top-level node name and the key associated with the top-level node.
- **AddChild.** Accepts the envelope of the child node, the child node name, and the key associated with the child node.
- **ChangeKey.** Accepts the node name and the key for that node.
- **SetXMLData.** Accepts the XML data used for the expression parser class.

Before the expression parser can be called, you must specify the top level, any children, and the associated XML data, for example:

```
var EP:ExprParser = new ExprParser;
EP.CreateTopLevel("Contact",strRecKey);
EP.AddChild("ListOfAccounts","Account",strChildKey);
EP.SetXMLData(xmlFileData);
var obj:Object = EP.ExpressionParser(strParseString);
```

If you want to change the child record, then call the ChangeKey function with the node name and key to be used to access the key, for example:

```
EP.ChangeKey("Account",strChildKey);
```

If you change the top-level key, then call the CreateTopLevel function again. This action clears the context and key arrays used for accessing the XML data.

Index