Siebel

Using Siebel Tools

August 2024

Siebel
Using Siebel Tools

August 2024

Part Number: F84091-05

Authors: Siebel Information Development Team

Contents

Preface

This preface introduces information sources that can help you use the application and this guide.

Using Oracle Applications

To find guides for Oracle Applications, go to the Oracle Help Center at *https://docs.oracle.com/*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program website*.

Contacting Oracle

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit *My Oracle Support* or visit *Accessible Oracle Support* if you are hearing impaired.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides! You can send an email to: *oracle_fusion_applications_help_ww_grp@oracle.com*.

1 What’s New in This Release

What’s New in This Release

This chapter tracks the changes in the documentation. It includes the following topics:

What’s New in Using Siebel Tools, Siebel CRM 24.8 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Deleting Developer Workspaces Using the siebdevcli Utility or Siebel Web Tools UI	Modified topic. It describes a new ability in Workspaces that allows for the deletion of developer Workspaces.

What’s New in Using Siebel Tools, Siebel CRM 24.7 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Deleting Developer Workspaces Using the siebdevcli Utility or Siebel Web Tools UI	New topic. It describes a new ability in Workspaces that allows for the deletion of developer Workspaces.

What's New in Using Siebel Tools, Siebel CRM 24.5 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Flattening Workspace Versions	Modified topic. Updates to the Flatten Workspace command line utility.

What's New in Using Siebel Tools, Siebel CRM 23.11 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Using the Script Profiler in Web Tools	New topic. The *Siebel Script Performance Profiler* is used with the eScript ST engine to observe and monitor the performance of a script.

What's New in Using Siebel Tools, Siebel CRM 23.7 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Viewing and Resolving Conflicts after Siebel Archive File Imports in Web Tools	New topic. You can view the imported objects and their properties, after the completion of RepositoryUpgrade or after you have imported a Siebel Archive File (SIF) through the command line interface.
Using the Command Line to Import an Archive	Modified topic. Importing an Archive can also be done using **`siebdevcli`** command line option.

What's New in Using Siebel Tools, Siebel CRM 23.5 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Favorites in Web Tools *Predefined Queries in Web Tools*	New topic. Favorites in Web Tools allow developers to quickly navigate to saved objects so that they can resume or inspect their progress. Predefined Queries allows developers to see only what they wish, reducing their development time.

What's New in Using Siebel Tools, Siebel CRM 22.7 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Supported Objects and Features *Unsupported Objects and Features*	Modified topics. The Task Editor and Validate Tool are now supported in Web Tools
Using the Project Drop-Down List *About the W Property* *Overview of Customizing Objects*	Modified topics. The note has changed in these topics since Tasks are now Workspace-enabled, which means you can edit them within a Workspace.
Avoid Rebase and Naming Conflicts for Workflow Processes and Tasks	Modified topic. This task applies also to tasks since tasks are now Workspace-enabled.
Open Inactive Conflicts	Modified topic. A new subsection (*About Overriding Name Conflicts for Tasks*) has been added to this topic. Resolving name conflicts is slightly different for Task UI.
Configuring Non-Workspace Objects	Modified topic. The Task and Task Group objects have been removed from the list of non-Workspace object types since tasks are now Workspace-enabled.
Comparing Workflow Processes or Task UIs	Modified topic. This topic was previously called *Comparing Different Versions of a Workflow Process or Task U*.
Controlling Workflow Process or Task UI Version Handling Comparing Different Versions of a Workflow Process or Task UI WF/Task Editor Toolbar	Obsolete topics. These topics have been removed from the guide.

What's New in Using Siebel Tools, Siebel CRM 22.4 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Avoid Rebase and Naming Conflicts for Workflow Processes and Tasks	New topic. Describes how to avoid a rebase and naming conflicts for Workflow Processes after an IRM/ Upgrade merge process.
Index Violation Errors	Modified topic. This topic has been expanded to describe how to delete unique indexes.
Validating Objects *Elements of the Validate Dialog Box* *Elements of the Validation Options Dialog Box*	Modified topics. The Validate Tool is now supported in Web Tools. This feature allows validation of objects (a single or multiple objects within an object type), Workspaces, or an entire repository.

Topic	Description
Single Object Type Validation	New topic. The Applet Menu in Web Tools has a new Validate option, which you can use to validate objects of a single object type.
Workspace Validation	New topic. The Workspace Toolbar in Web Tools has a new Validate option, which you can use to validate Workspaces.
Batch Validation	Modified topic. Batch validation has been enhanced to restrict the volume of objects to validate (by using the exclusions defined in the Advanced Options dialog in the Web Tools client). **Note:** This topic was previously called *Using the Command Line to Validate Objects*.

What's New in Using Siebel Tools, Siebel CRM 22.1 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Debugger Service	Modified topic. Information about the (DbgProcCfg subsystem) Host parameter has been updated. The subtopic about Debugger Service Multi-Instances Support is new.
Enabling Script Editor and Server Script Debugger in Web Tools	Obsolete topic. This topic has been removed from the guide. The Script Editor and Debugger feature is enabled by default.

What's New in Using Siebel Tools, Siebel CRM 21.11 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Workspace Types and Inheritance *Basic Workspace Structure* *Workspace Best Practice Guidelines* *Workspace Limitations*	New topics. These topics – describing Workspace types, inheritance, structure, limitations, and guidelines for managing Workspaces – are new for 21.10 Update release.
How LOVs Work in Workspaces	New topic. Describes how LOVs work in Workspaces.
Delivering Workspaces	Modified topic. Workspaces must be delivered sequentially, rather than in parallel.
Exporting Objects to an Archive *Exporting Workspaces to an Archive*	Modified topics. There is a new Archive menu in Web Tools.

Topic	Description
Importing Objects From an Archive	New topic. Importing objects from an archive is now supported in Web Tools.

What's New in Using Siebel Tools, Siebel CRM 21.9 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
About Inspecting Objects in Workspaces	New topic. You can inspect objects in a Workspace to test their functionality prior to delivering them to the MAIN Workspace. Doing this requires objects to be compiled in-memory at run time, which in turn impacts performance.
Creating New Workspaces	Modified topic. A full stop or period is allowed in the name of a Workspace.
Delivering Workspaces	Modified topic. The logging location has changed for delivering a Workspace.
Debugger Service	Modified topic. The Debugger Service can only be hosted currently on one Siebel Server.
Overview of Exporting and Importing Repositories	Modified topic. While you use the Database Configuration Wizard to export or import entire repositories, you use the database utilities that your RDBMS vendor provides to back up the entire contents of the Siebel database.
Overview of Archiving Objects	Modified topic. Archiving is not supported nor intended for the migration of entire repositories from one environment to another.

What's New in Using Siebel Tools, Siebel CRM 21.7 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Using Workspaces	Modified topic. When you create a List of Value (LOV) in an Integration Workspace (IWS), the LOV is automatically created in every IWS, whether existing or new, but will be inactive in those Workspaces.
Flattening Workspace Versions	Modified topic. You must flatten Workspaces before running an MLOV conversion.
Deleting Repositories	Modified topic. Added `-d` (DB Platform Name) to the list of RRCleanup utility parameters.
Using the Command Line to Export Workspaces to an Archive	New topic. You use the siebdevcli command line to export whole Workspace objects or any particular object under a Workspace to an archive.

What's New in Using Siebel Tools, Siebel CRM 21.4 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Using the New Object Wizard	Modified topic. The New Object Wizard is only partially supported in Web Tools - that is, to create a new Web service object. The ability to create a new General, Applet, or Task object using the New Object Wizard is currently not supported in Web Tools.
Creating a New Web Service Object in Web Tools	New topic. The Web Service Wizard is now available in Web Tools.
Workspace Dashboard	Modified topic. Describes the Workspace Dashboard.
Executing the Full Publish Process	Modified topic. Added information about when to use and how to use `FullPublishInternal`.
Enabling Workspaces	New topic. It is only necessary to run the EnableWorkspace utility, to manually re-enable Workspaces, if you execute a FullPublish operation on the repository.

What's New in Using Siebel Tools, Siebel CRM 21.3 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Workspace Delivery When Deploying Multiple Languages	New topic. Added information about Workspace delivery when deploying multiple languages.

2 Siebel Tools Setup

Siebel Tools Setup

This chapter describes how to set up Oracle's Siebel Tools. It includes the following topics:

- *About Siebel Tools*
- *Roadmap for Setting Up and Using Siebel Tools*
- *Process of Setting Up the Development Environment*

About Siebel Tools

Siebel Tools is an integrated development environment that you can use to customize Siebel CRM. You can use it to modify predefined Siebel objects or to create custom objects that meet your business requirements. For example, you can use Siebel Tools to modify the data model, modify business logic, or customize the user interface that the Siebel client shows. Siebel Tools allows you to develop a single configuration that you can use to do the following:

- Deploy Siebel CRM across multiple types of clients
- Support multiple Siebel Business Applications and languages
- Upgrade Siebel product releases
- Maintain Siebel CRM

Declarative configuration is a type of programming technique that uses objects and object properties in the Siebel repository to implement the logic that your business requires. Siebel Tools uses declarative configuration to create and modify the object definitions that define a Siebel application. Siebel Tools is not a programming environment. You do not modify the source code or write SQL. Siebel CRM uses the terms object and object definition differently than developers who use programming languages that use similar terms, such as object, object class, or object instance. For more information about the Siebel repository, see *Managing Repositories* For more information about the objects, object definitions, and the object hierarchy that Siebel CRM uses, see *Configuring Siebel Business Applications* .

Note: The *Siebel Bookshelf* is available on Oracle Technology Network (http://www.oracle.com/technetwork/indexes/documentation/index.html) and Oracle Software Delivery Cloud. It might also be installed locally on your intranet or on a network location.

Roadmap for Setting Up and Using Siebel Tools

To set up and use Siebel Tools, perform the following tasks:

1. *Process of Setting Up the Development Environment*

2. *Creating and Modifying Objects*
3. *Validating Objects*
4. *Testing and Troubleshooting Your Customizations*
5. *Exporting Objects to an Archive*

Process of Setting Up the Development Environment

To set up the development environment, do the following tasks:

1. *Installing Siebel Tools*
2. *Setting the Language Mode*
3. *Controlling Confirmation Dialog Box Display*
4. *Resetting Development Tools Options*

This process is a step in *Roadmap for Setting Up and Using Siebel Tools*.

Installing Siebel Tools

In this guide, $SIEBEL_HOME represents the folder where you install Siebel Tools. For example, a typical location would be `C:\Siebel\Tools`.

For more information about:

- How to install Siebel Tools, see the *Siebel Installation Guide* . For example, see *Siebel Installation Guide for Microsoft Windows* .
- System requirements, such as supported versions of Microsoft Windows, see *Siebel System Requirements and Supported Platforms* on Oracle Technology Network.

  Note: For Siebel CRM product releases 8.1.1.9 and later and for 8.2.2.2 and later, the system requirements and supported platform certifications are available from the Certifications tab on My Oracle Support.

This task is a step in *Process of Setting Up the Development Environment*.

Configuring Workflow Process or Task UI Version Handling

Note: This task applies only to Siebel Tools.

Depending on value of the Status property, you can control how versions of a Workflow Process or task UI that you open with an editor are handled. The following procedure shows how to configure the options for handling Workflow Process or task UI versions.

This task is a step in *Process of Setting Up the Development Environment*.

To control Workflow Process or task UI version handling

1. Select Options from the View menu.
2. In the Development Tools Options dialog box, select the General tab.

3. In the Workflow and Task Configurations section, configure the options as required. The following table describes the available options.

Option	Description
Automatic Revision in WF/Task Editor and Version Check	Select or deselect the check box as required. ◦ **If Selected** and if you use an editor to open a Workflow Process or task UI and the Status property on the Workflow Process or task UI is Completed, then a copy of the Workflow Process or task UI is created, the status on the copy is set to In Progress, and an editable version of it is opened in the editor. ◦ **If Deselected**, then the editor opens and displays a version of the Workflow Process or task UI that you cannot edit.
Automatically Close All the Previous WF/Task Versions	Select or deselect the check box as required. ◦ **If Selected** and if you use an editor to open a Workflow Process or task UI and the Status property on the Workflow Process or task UI is Completed, Not In Use, or Expired, then any prior versions of the Workflow Process or task UI are closed. ◦ **If Deselected**, then any prior versions of the Workflow Process or task UI are not automatically closed.

4. Click OK.

Setting the Language Mode

Note: This task applies only to Siebel Tools.

The *language mode* is a type of mode that allows you to configure Siebel CRM to display text in a language other than English. For example, the German (DEU) language mode allows you to view text in some objects of Siebel Tools in German, such as the text that a string contains or object definitions that the Object List Editor shows. The language mode determines the set of locale data that Siebel Tools uses when it compiles the repository, and during checkin and checkout. If you add a language to the Siebel database that does not come predefined with Siebel CRM, then this language must use all capital letters. For more information, see *About Predefined Objects* and *Siebel Global Deployment Guide* .

This task is a step in *Process of Setting Up the Development Environment*.

To set the language mode

1. Make sure the repository includes the language data that Siebel Tools must show.
2. Select Options from the View menu.
3. In the Development Tools Options dialog box, select the Language Settings tab.
4. In the Tools Language Mode section, select a value from the Language drop-down list, and then click OK.
5. (Optional) To enable language override, make sure the Enable and Use Language Override check box is selected (includes a check mark). For more information, see the following subtopic about Enabling Language Override.
6. Click OK.

Enabling Language Override

A *language override* is a non-translatable, locale property that you can configure differently for different locales. For example, you can configure Siebel CRM to use a specific height for the address field in French and to use a different height in English. If you enable language override, then Siebel CRM might create more locale records in the repository. To avoid unnecessary records, it is recommended that you enable language override only if your deployment requires it. For more information, see *Configuring Nontranslatable Locale Object Properties* and *Configuring Siebel Business Applications* .

Controlling Confirmation Dialog Box Display

Note: This task applies only to Siebel Tools.

You can control whether Siebel Tools displays a confirmation dialog box or not. The following procedure shows how to configure whether Siebel Tools display a confirmation dialog box or not.

This task is a step in *Process of Setting Up the Development Environment*.

To control the display of the confirmation dialog box

1. Select Options from the View menu.
2. In the Development Tools Options dialog box, select the General tab.
3. In the Editing Confirmation Dialogs section, select the check box next to one of the following options as required:
    - **Display a confirmation dialog box.** Select the check box to enable the display of the confirmation dialog box.
    - **Do not display a confirmation dialog box.** Deselect the check box (to enable the display of the confirmation dialog box).
4. Click OK.

Resetting Development Tools Options

Note: This task applies only to Siebel Tools.

If Siebel Tools behavior is not consistent with the preferences you set, then you can reset them.

This task is a step in *Process of Setting Up the Development Environment*.

To reset development tools options

1. Do one of the following:
    a. Reset preferences in the Development Tools Options dialog box:
        - Select Options from the View menu.
        - In the Development Tools Options dialog box, go to the various tabs and reset preferences as required.

   - Click OK.

b. Delete the file that stores your preferences:

   - Close Siebel Tools.
   - Navigate to the `$SIEBEL_HOME\BIN` folder and delete the `devtools.prf` file.
   - Log back in to Siebel Tools and select Options from the View menu.
   - Set preferences in the Development Tools Options dialog box.

The preferences that you set are stored in the Development Tools Options dialog box in the `devtools.prf` file. If you delete this file, then all your preferences will be reset to their default values.

3 Web Tools Setup

Web Tools Setup

This chapter describes how to set up Oracle's Web Tools. It includes the following topics:

- *About Web Tools*
- *Setting Up Web Tools*

About Web Tools

Oracle is gradually migrating from Siebel Tools to Web Tools to simplify and expedite the process of configuring Siebel Business Applications. Web Tools is a Web-enabled application that runs on a server, and you access it in a browser by using a Siebel Open UI client. Siebel Tools is a stand-alone application that runs on Microsoft Windows.

Note: Web Tools does not support thick client.

In Web Tools, you currently have access to some of the functionality in Siebel Tools, so you can use Web Tools to perform some of the tasks that in the past you performed in Siebel Tools. In both Siebel Tools and Web Tools, you need to be added to the database to access these tools. In addition, for Web Tools, you need to have the Composer Administrator responsibility to access the *Workspace Dashboard* to perform developer or configuration operations. To learn more about Workspaces in Web Tools, see *Setting Up Web Tools*.

Using Web Tools, note that:

- You must ensure that Workspaces are enabled on the environment before you work on Web Tools. For more information about how to use and administer Workspaces, see *Using Workspaces*.
- Web Tools has the same client requirements as other Siebel Open UI clients. For more information, see *Deploying Siebel Open UI* .
- Web Tools uses its own theme, which cannot be configured. For themes that can be configured in Siebel applications, see *Configuring Siebel Open UI* .

Benefits of Web Tools

Web Tools provides the following benefits:

- More control in the client.
- More elements for composition of the user interface.

- More efficient rendering of Web pages.

  The DOM (Document Object Model) structure in Siebel Business Applications is simplified and the DOM is assembled using a single HTTP response, not multiple relays, between the client and the server.

  When you use Siebel Tools, content is trimmed only from the client. Also, post-response JavaScript execution trims content. When you use Web Tools, the server evaluates client capabilities, such as resolution and form factors. Content is tailored from the server, and more efficient options are available to manage content.
- Decreased network traffic.

  Only the content that is necessary to currently render the user interface is included in responses. Extraneous content that the client must later discard or ignore is not included in responses. This decrease in content decreases the network traffic.
- Improved file loading.

  The framework files that are needed to start a Siebel application are loaded from the manifest. The files for the presentation model and physical renderer that are needed to render Web pages are also loaded from the manifest.

Supported Objects and Features

The following features are available and supported in Web Tools:

- Locking and unlocking projects in the local repository for non-Workspace objects.
- Configuring symbolic strings.
- Adding to archive.
- Sorting on objects.
- List of Values, Analytic Strings, and System Preferences.
- Using Object Explorer and Object List Editor.
- Using Siebel IDE for layout editing.
- IO deploy/undeploy.
- Using the Siebel Script Editor, the eScript Engine, and the Siebel Debugger.
- Setting debug options.
- Workflow Editor.
- Task Editor.
- Validate Tool.
- Validating objects, Workspaces, tasks, and repositories.

Note: For more information about Workflow Processes and using the Workflow Editor, see *Siebel Business Process Framework: Workflow Guide* . For more information about Task UI and using the Task Editor, see *Siebel Business Process Framework: Task UI Guide*

The following objects are available and supported in Web Tools:

- Applet

- Application
- Business Component
- Assignment Attribute
- Assignment Criteria
- Bitmap Category
- Business Object
- Business Service
- Class
- Command
- Content Object
- DLL
- Dock Object
- EIM Interface Table
- Find
- HTML Hierarchy Bitmap
- Icon Map
- Import Object
- Integration Object
- Link
- Menu
- Message Category
- Pick List
- Project
- Repository
- Schema Maintenance Phase
- Schema Maintenance Process
- Schema Maintenance Step
- Screen
- Symbolic String
- System Activity Object
- Table
- Task
- Task Group
- Toolbar
- Type
- View
- Web Page

- Web Template
- Workflow Policy Column
- Workflow Policy Object
- Workflow Policy Program
- Workflow Process

CAUTION: Operations are not allowed on the following read-only objects: Dock Object, the Schema Maintenance objects, Type, Repository, Table, and EIM Interface Table.

Unsupported Objects and Features

The following features are currently not supported in Web Tools:

- Various windows and wizards (such as, Synchronize).
- Tagging objects to manage developer changes.
- Examining and comparing objects.
- Using the Script Profiler.
- Configuring locale data and using the Locale Management utility to manage locales.
- Using Application Deployment Manager (ADM).

The following objects are currently not supported in Web Tools:

- Entity Relationship Diagram
- Help Id
- Pager Object
- Server Component Type
- Search Category
- Search Engine
- Search Index

Setting Up Web Tools

This section includes information about the setup tasks that you must perform to deploy Web Tools for a new deployment or for an existing deployment that you originally installed for an earlier version. It includes the following topics:

- *Manually Running the Migration Script*
- *Troubleshooting and Verifying the Web Template Migration Process*
- *Publishing Changes to Siebel Web Templates*
- *HTML Tags Migrated to Web Tools*

- *Component Parameters for Siebel Business Applications*

Tip: Ensure that you enable the applicable component group while configuring the Siebel Server. For more information about configuring the components, see *Siebel Installation Guide* and *Siebel Applications Administration Guide* .

Manually Running the Migration Script

Enter the following command at the command-line interface to manually run the migration script for the Web template migration process:

```
-jar $DbsrvrRoot\common\SWTClob.jar /s $SiebelRoot /c "$ODBCDataSource"
/t $TableOwner /u $TableOwner /p $TablePassword /o $SiebelLogDir /d $DatabasePlatform
/r $RepositoryName /j $WebTemplatesDir /w $WSUSerName /x $WorkspaceName /b $BranchedWS
/i $WebTemplateName
```

Where:

- $DbsrvrRoot is the path of the database server installation.
- $SiebelRoot is the path of the Siebel Server installation.
- $SiebelLogDir is the path to the log directory.
- $DatabasePlatform is the database platform such as Oracle, DB2UDB, MSSQL, and DB2390.
- $RepositoryName is the Name of the Repository such as Siebel Repository.
- $WebTemplatesDir is the directory for Web Templates such as `$SiebelRoot/Webtempl`.
- $WebTemplateName is the list of template names for Incremental Migration. If not passed, all Web Templates are considered for Migration.

The following are optional parameters applicable only for Workspace Environment:

- $WSUserName is the Workspace Owner user such as SADMIN.
- $WorkspaceName is the Workspace Name to be created such as `dev_xxxx_xxx`.
- $BranchedWS is the Workspace Branch under which $WorkspaceName needs to be created such as MAIN.

Note: For Workspace-enabled environment, the New Private Workspace branch will be created and then delivered to the Integration or MAIN branch through the defined delivery process. The delivery process will internally publish the modified content into the Runtime Repository.

Troubleshooting and Verifying the Web Template Migration Process

After the migration script runs, you can access the SWTClob.log file to review the results of the Web template migration process. This file is available when you manually run the migration script and when the migration script automatically runs during upgrade. The log file will list the probable errors, if any, or will state the program completion message. Look for messages in the log that indicate incomplete or partial conversion or failure in conversion:

- **Incomplete conversion for <template name>.** This error indicates that the template file still contains some unconverted od tags. The possible reason for this is that occurrences of SWE markup in the template file are not as per their documented usage. To resolve, review the remaining od tags in the converted template.

- **Unable to convert <template name>.** This error indicates that the template could not be converted at all. The possible reason for this is an internal conversion error when you tried to migrate to ODH. To resolve, review the template content for any inconsistency.

  > **Note:** You can ignore the Malformed markup log that indicates that the template does not have a complete markup. It relates to tags that are not closed in the same file at the correct level.

Publishing Changes to Siebel Web Templates

The Web template migration process generates the table-based content for Siebel Web templates. If the migration installation case applies to you, then you must publish the content in these tables so that the content is implemented in the Siebel application. If the new installation case applies to you, then publication of this content is automatically completed for you.

To publish the table-based content for Siebel Web templates, enter the following command in a command window on the computer where you installed Web Tools:

```
siebdev /c tools.cfg /u $SIEBUSER /p $SIEBPWD /d "$ODBCDataSource" /l enu /
fullpublish
```

HTML Tags Migrated to Web Tools

The following table describes the HTML (Hypertext Markup Language) tags that are migrated to fields in Web Tools during the Web template migration process.

Tag Name	General Tag Purpose
div	Generates a container record for a Web template (div) item and for its properties.
li	Generates a container record for a Web template (`li` or list) item and for its properties.
span	Generates a container record for a Web template (span) item and for its properties.
ul	Generates a container record for a Web template (`ul` or unordered list) item and for its properties.

Component Parameters for Siebel Business Applications

The following table describes some key component parameters for Siebel Business Applications. For information about how to change component parameters, see *Siebel System Administration Guide* .

Parameter	Description

EnableSafeBoot	Siebel OOTB Runtime Repository is shipped as a loadable unit (DLL/SO). When the EnableSafeBoot parameter is set to TRUE, the OOTB Runtime Repository is used.
	
	• For Mobile Web Client, set the EnableSafeBoot parameter to TRUE along with the RepositoryType parameter to RUNTIME in the respective configuration file.
	
	To set EnableSafeBoot and RepositoryType in the Siebel Mobile Web Client.
	a. Use a text editor to open the respective configuration file.
	b. Set the EnableSafeBoot parameter to the following value:
	`EnableSafeBoot=TRUE`
	c. Set the RepositoryType parameter to the following value:
	`RepositoryType=RUNTIME`
	The EnableSafeboot parameter should be added to the InfraObjMgr section of the configuration file, while the RepositoryType parameter is available in the InfraObjMgr section by default.
	
	Note: For Mobile Web Client, the EnableSafeBoot parameter is used only for troubleshooting purpose. If Mobile Web Client fails to load after repository changes, then set the EnableSafeBoot parameter to TRUE for the Mobile Web Client and try again. After troubleshooting, you can revert the value of the EnableSafeBoot parameter.
	
	• For Siebel Enterprise Server, set EnableSafeBoot as follows:
	`change param EnableSafeBoot=TRUE for comp SCCobjMgr_enu`
	
	Note: Web Tools runs in safe boot mode by default so changing the value of EnableSafeBoot will not affect the application in any way.

ManifestSafeLoad	ManifestSafeLoad can be set to TRUE or FALSE as follows: • If set to TRUE, the application starts with only the Oracle provided manifest records (that is, seeded records in the manifest). • If set to FALSE (the default value), the application starts with both custom-configured and seeded records in the manifest. Change the parameter as follows: `change param ManifestSafeLoad=TRUE for comp SCCObjMgr_enu` **Note:** Changing the value of ManifestSafeLoad will not affect the Web Tools application in any way.
DefaultNavigation	This parameter allows the application to start with the preferred navigation control. The available options are: NAVIGATION_SIDE, NAVIGATION_TREE, or NAVIGATION_TAB. Change the parameter as follows: `change param DefaultNavigation=NAVIGATION_TREE for comp SCCObjMgr_enu` **Note:** Changing the value of DefaultNavigation will not affect the Web Tools application in any way.

4 Getting Started With Your Application

Getting Started With Your Application

This chapter describes how get started with your application. It includes the following topics:

About the Object Explorer and Object List Editor

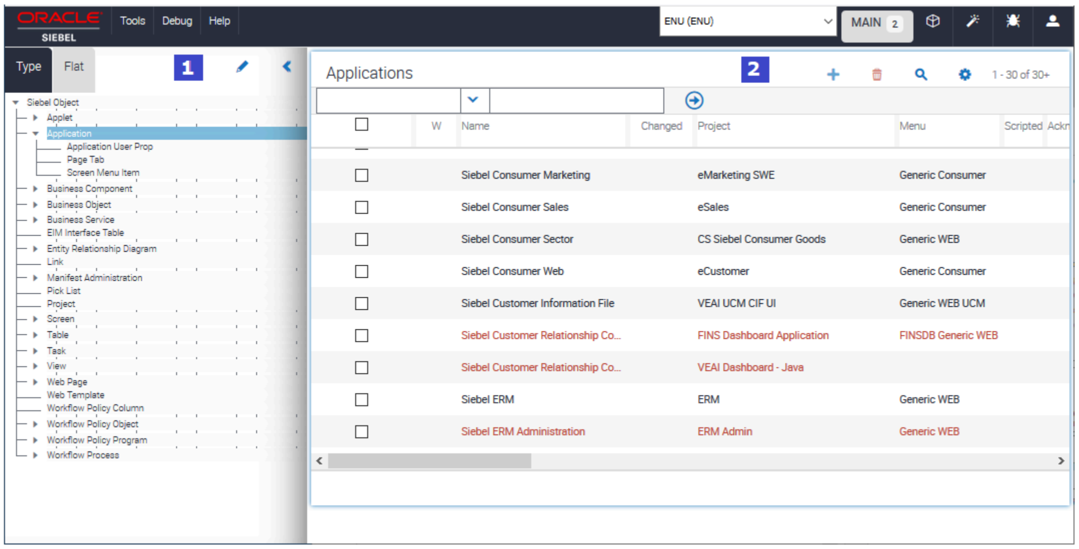
The *Object Explorer* is a window that displays the Siebel object hierarchy. It includes a tree structure which facilitates navigation through the object types that the hierarchy contains.

The *Object List Editor* is a window that displays the object definitions of the object type that is selected in the Object Explorer. You can view and modify object definitions in the Object List Editor window.

For more information about the object hierarchy that Siebel CRM uses, see *Configuring Siebel Business Applications* .

The following image illustrates the Object Explorer and Object List Editor in Web Tools. A similar Object Explorer and Object List Editor is available in Siebel Tools:

1. **Object Explorer.** In this image, the Application object type is selected in the Object Explorer.
2. **Object List Editor.** In this image, the Applications list appears (and corresponding object definitions) in the Object List Editor. For more information, see *Using the Object List Editor*.



- Siebel Object
  - Applet
  - Application
    - Application User Prop
    - Page Tab
    - Screen Menu Item
  - Business Component
  - Business Object
  - Business Service
  - EIM Interface Table
  - Entity Relationship Diagram
  - Link
  - Manifest Administration
  - Pick List
  - Project
  - Screen
  - Table
  - Task
  - View
  - Web Page
  - Web Template
  - Workflow Policy Column
  - Workflow Policy Object
  - Workflow Policy Program
  - Workflow Process

☐	W	Name	Changed	Project	Menu	Scripted	Ackn
☐		Siebel Consumer Marketing		eMarketing SWE	Generic Consumer		
☐		Siebel Consumer Sales		eSales	Generic Consumer		
☐		Siebel Consumer Sector		CS Siebel Consumer Goods	Generic WEB		
☐		Siebel Consumer Web		eCustomer	Generic Consumer		
☐		Siebel Customer Information File		VEAI UCM CIF UI	Generic WEB UCM		
☐		Siebel Customer Relationship Co...		FINS Dashboard Application	FINSDB Generic WEB		
☐		Siebel Customer Relationship Co...		VEAI Dashboard - Java			
☐		Siebel ERM		ERM	Generic WEB		
☐		Siebel ERM Administration		ERM Admin	Generic WEB		

An *object type* is an entity that includes a predefined set of properties. You can use it as a template to create an object definition. An application is one kind of object type. For more information, see *About Predefined Objects*.

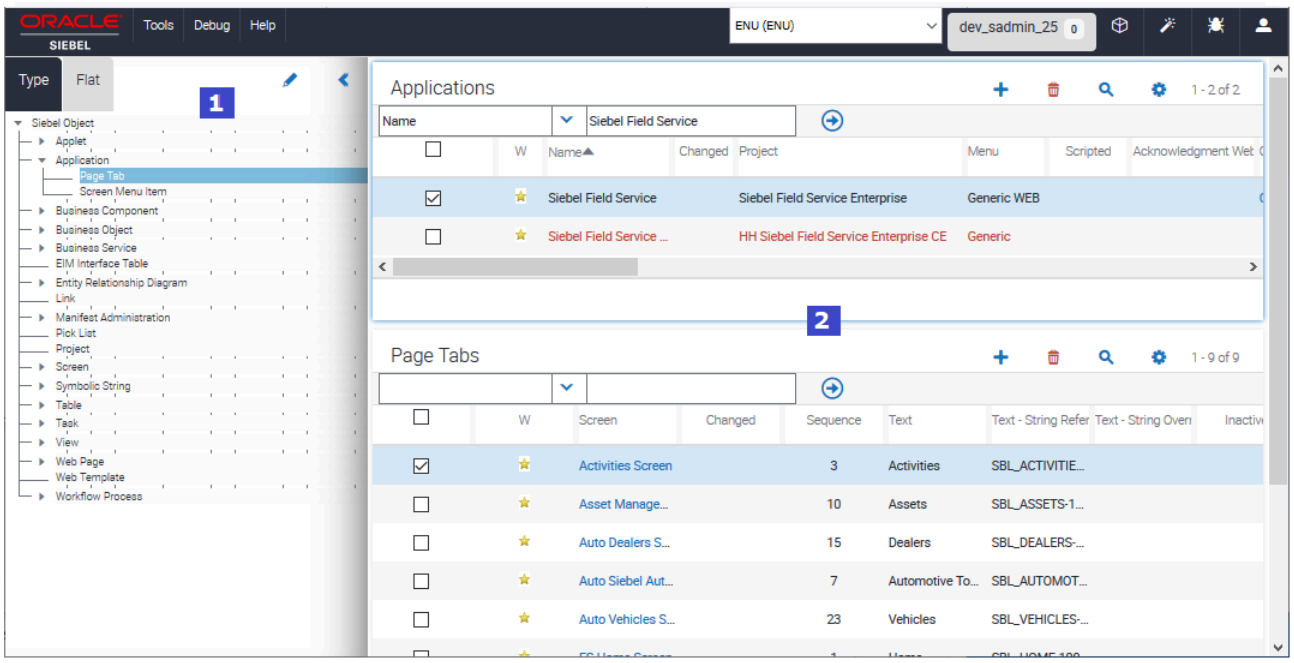
An *object definition* implements one piece of the software. This object definition consists of *object properties*, which are characteristics of this piece of the software. The Object Explorer and Object List Editor display all the object definitions that the repository contains. Siebel Field Service is an example of an object definition of an application. Siebel CRM uses the terms *attribute* and *property*. These terms have the same meaning.

If an object type is not visible in the Object Explorer, then you can display it. For more information, see *Displaying Object Types in the Object Explorer*.

Siebel Object Hierarchy

The following image illustrate the *object type hierarchy* and *object definition hierarchy* for a parent and child relationship, which is a type of hierarchical relationship between one object type and another object type. In the image, the Page Tabs object definitions for the Siebel Field Service application are shown as follows:

1. **Object type hierarchy (Object Explorer).** In this example, the Page Tab object type and the Screen Menu Item object type are child objects of the parent Application object type. The Object Explorer displays this relationship as a hierarchical tree that you can expand or collapse. For more information, see *Using the Types Tab*.
2. **Object definition hierarchy (Object List Editor).** The Object List Editor uses two windows to display this parent and child relationship. In this example, the Applications list displays the object definition for the parent Siebel Field Service application. The Page Tabs list displays the object definitions of the child page tabs that the repository contains for the Siebel Field Service application.



Applications 1 - 2 of 2

Name | Siebel Field Service

	W	Name	Changed	Project	Menu	Scripted	Acknowledgment Wel
☑	★	Siebel Field Service		Siebel Field Service Enterprise	Generic WEB		
☐	★	Siebel Field Service ...		HH Siebel Field Service Enterprise CE	Generic		

Page Tabs 1 - 9 of 9

	W	Screen	Changed	Sequence	Text	Text - String Refer	Text - String Overr	Inactive
☑	★	Activities Screen		3	Activities	SBL_ACTIVITIE...		
☐	★	Asset Manage...		10	Assets	SBL_ASSETS-1...		
☐	★	Auto Dealers S...		15	Dealers	SBL_DEALERS-...		
☐	★	Auto Siebel Aut...		7	Automotive To...	SBL_AUTOMOT...		
☐	★	Auto Vehicles S...		23	Vehicles	SBL_VEHICLES-...		

About Predefined Objects

This guide uses the term *predefined* to describe an object or configuration that comes already defined when you first install Siebel CRM, Siebel Tools, or Web Tools. Each object that the Object List Editor shows after you install Siebel, but before you make any modification, is a predefined object. A *custom object* is a new object that you create. A *modified object* is a predefined object or custom object that you modified.

Using the Right-Click Menu

Note: Right-click is available only in Siebel Tools.

A *right-click menu* is a type of context-sensitive menu that appears if you right-click an object or an element. It allows you to do the following:

- Create, copy, or delete a record.
- Undo modifications that you make to a record.
- Display the name and status of a toolbar. You can right-click any toolbar. You can also customize how a toolbar appears.
- Lock an object.
- Add an object to an archive.
- Access wizards.

Displaying Object Types in the Object Explorer

This topic describes how to display object types in the Object Explorer.

To display object types in the Object Explorer

1. In the Object Explorer, click Edit (the pencil icon).
2. Scroll down through the Object Explorer Hierarchy list until you locate the object type(s) you want to display.
3. Select the check box next to each object type that you want to appear in the Object Explorer.

   If you add a check mark to a top-level object type, such as Applet, then a check mark is also added to all Applet child objects. You can expand the parent tree to display only some child objects. The object types that the Object Explorer shows immediately after you install Web Tools, but before you expand any object types in the Object Explorer, are top-level object types.
4. (Optional) To restore the Object Explorer to the default settings when you first installed Web Tools, click Refresh.
5. Click Save.

Displaying Hidden Object Types and Properties

This topic describes how to display hidden object types and properties. A *hidden object type* is an object type that Siebel CRM does not show in the client. For more information, see *Siebel Object Types Reference* .

To display hidden object types and properties

1. Close your Siebel application.
2. Use a text editor to open the `tools.cfg` file.
3. In the Siebel section, set the value of the following parameter to `All`:

   ```
   ClientConfigurationMode
   ```
4. Save the file, and then restart your Siebel application.

Using the Project Drop-Down List

The Object Explorer includes the Project drop-down list that allows you to filter objects according to project. The following procedure shows how to use the Project drop-down list so that the Object Explorer displays only the object types that reference the Account project.

Note: In the Object Explorer in Web Tools, most of the object types are Workspace-enabled, except for the following: Table, Repository, Type, EIM Table, and Project. For more information on Project behavior in Web Tools, see *Using Workspaces*.

To use the Project drop-down list

1. In the Object Explorer, click the Project drop-down list.
2. Choose Account.

You can choose All Projects in the Project drop-down list to reset the Object Explorer so that it displays objects types for all projects.

Using the Types Tab

The Object Explorer displays the Types tab by default, which you use to navigate the object hierarchy. The Object Explorer does not display all object types by default. For more information, see *Displaying Object Types in the Object Explorer*. For more information about the object hierarchy, see *Configuring Siebel Business Applications* .

To use the Types tab

1. In the Object Explorer, click the Types tab.

   The top-level object types appear in alphabetical order.
2. To display child object types, expand the tree for an object type by clicking the plus (+) sign next to the object type.
3. To collapse an object type, click the minus (–) sign.

   In Web Tools, click the arrow sign to expand and collapse the Object Explorer tree.

Using the Detail Tab

Note: The Detail tab is available only in to Siebel Tools.

The Object Explorer includes a Detail tab that allows you to view and expand an object type. When you select the Detail tab, the object properties for the selected object appear in the Object List Editor

Using the Flat Tab

The Object Explorer includes a Flat tab that displays all object types in a single, alphabetic list. It does not display a hierarchy. You can use it to do the following:

- Find a child object with an unknown parent. For example, you create a new field but do not remember the business component that this field references. You can click the Flat tab, click the Field object type, and then query the Name property for the field name. Each record that is returned includes a parent property that displays the business component name.
- Determine how Siebel CRM typically uses objects and properties, such as how it uses a predefault value or the format that it uses in a calculated field.

To use the Flat tab

1. In the Object Explorer, click the Flat tab.

   The Object Explorer displays a list of all the object types in the repository. It displays this list in alphabetic order and with no hierarchy.
2. In the Object Explorer, click Field.

   The Object List Editor displays all the object definitions in the repository for Field. This example describes how to use the flat tab to identify all the business components that include a field named Account Address.
3. In the Fields list, query the Name property for Account Address.

   The Object List Editor displays a list of all the field object definitions in the repository with the name Account Address.
4. Click the Fields Menu (the cogwheel icon) on the applet banner and select Columns Displayed (or press CTRL +SHIFT+K).
5. In the Columns Displayed dialog box, select Parent Business Component from Available Columns and move it into Selected Columns, and then click OK.

   The Parent Business Component column appears in the Object List Editor.
6. In the Parent Business Component property, click Order Entry - Orders.

   The Object List Editor displays the object definition for the Order Entry - Orders business component.

Getting Documentation on an Object

This topic describes how to get documentation about an object.

To get documentation about an object

1. In the Object Explorer, select an object type such as Business Component.
2. Press the F1 key.

   Siebel Tools Online Help opens with documentation about the selected object type, including information about its properties. The content in this online help also appears in *Siebel Object Types Reference* .

Using the Object List Editor

You can use the Object List Editor to edit the properties of an object definition. For example, if you choose the Application object type in the Object Explorer, then the Object List Editor displays a list of all the applications that the repository contains. One row in the Object List Editor represents one object definition. For example, the values that the Object List Editor shows in the properties of the Siebel Field Service application represent one object definition.

The Object List Editor displays each object property as a column in the list. Information that you enter in each column represents a property value. You can modify the property values in an object definition. You cannot modify the set of properties that constitute an object definition.

Locating Object Definitions in the Object List Editor

The following procedure shows how to use the Object Explorer and Object List Editor to locate and modify the Comments property of the Siebel Field Service application, and then examine the page tabs that this application uses. For information about locating object definitions for multiple object types, see *Searching the Repository*.

To locate object definitions in the Object List Editor

1. In the Object Explorer, click Application.

   For more information, see *About the Object Explorer and Object List Editor*.
2. In the Applications list (in the Object List Editor), query the Name property for Siebel Field Service.

   For more information, see *Using the Query Menu to Run a Query*.

   If the object definition exists, then the Object List Editor displays it in the Applications list. Querying for a record in this way returns a single, isolated record. This technique helps to make sure that you select the correct object definition and that it remains selected while you use the Object Explorer and Object List Editor to navigate the object hierarchy, or if you do other work, such as delivering or revising. It reduces the possibility that you might mistakenly modify another object definition. For more information about object types and object definitions, see *About the Object Explorer and Object List Editor*.
3. Locate the property you want to modify and then type in a new value for the property, select a value from the drop-down list, or add a check mark to the check box.

   For example, locate the Comments property and type in the following text:

   ```
   This is the object definition for the Siebel Field Service application.
   ```

   For more information, see *Using the Object List Editor*.
4. To save your modifications, click anywhere outside of the row.
5. In the Object Explorer, expand the Application tree, and then click Page Tab.
6. Review the records that the Object List Editor shows in the Page Tabs list.

   For more information, see *Siebel Object Hierarchy*.

About the W Property

If an object is locked and editable, then a pencil icon appears in the W property in the Object List Editor. The pencil icon in *Siebel Object Hierarchy* indicates that all objects in the Object List Editor are locked and editable. If you lock a parent object, then all child objects of this parent are locked throughout the hierarchy. If you lock a child object, then all objects that are parents of this child are locked throughout the hierarchy.

Note: Locking a project is still required before you edit the non-Workspace objects. The non-Workspace objects are Table, Repository, Type, EIM Table, Projects, Dock objects, Schema Maintenance, and Server Components. For more information, see *Using Workspaces*.

About the Changed Property

If you edit a record, then Siebel adds a check mark (or Y) to the Changed property. This check mark indicates that you modified this object definition since a specific date and time. If the Changed property does not include a check mark (or includes N), then no modification has occurred since the specified date and time. If you modify a child object, then Siebel also adds a check mark (or Y) to the Changed property of all objects that are parents of this child in the hierarchy.

To configure the Changed property

1. Click Settings (the photo icon) on the application-level toolbar and then click Settings again.
2. On the Settings form that appears, enter the date and time by which updated records need to be marked as changed in the following field: *Mark as changed, all records that have been updated since: <MM/DD/YYYY/ HrMinSec>*.

   If a modification occurs on a record that the Object List Editor shows and if this modification occurs:

   - **On or after the date you specify,** then Y appears in the Changed property.
   - **Before the date you specify,** then N appears in the Changed property.

About Links

If the property value in the Object List Editor references the name of another object, then this value appears as a link that you can click. If you click this link, then the object definition of the item you clicked opens.

To be able to use links, make sure that you are assigned the Developer responsibility in the Administration - Application screen, Responsibilities view in the Siebel client. For more information, see *Siebel Security Guide* .

About the Inactive Property

If a record's Inactive property is TRUE or Y, then the record is deactivated in the repository the next time you deliver your modifications. For more information on how to deliver your Workspace, see *Delivering Workspaces*.

If an object definition becomes obsolete due to an update or due to a new requirement, then do not delete the unused objects. Instead, it is recommended that you set the Inactive property of this object definition to TRUE. Siebel CRM does not reference an inactive object.

Defining the Name of the User Property Object Type

It is not necessary to enter the name of a valid user property if you add a user property to an object, such as an applet or business component. Instead, you can use a drop-down list in the Name field of the user property. Siebel then displays a dialog box that lists the valid user properties that you can define for the object. For more information about user properties, see *Siebel Developer's Reference* .

Selecting Multiple Records in the Object List Editor

You can use multiselect to select more than one record in the Object List Editor.

To select multiple records in the Object List Editor

1. To select multiple records, that are not consecutive: Hold down the CTRL key while you click each record.
2. To select multiple consecutive records: Click a record, hold down the SHIFT key, and then click another record.

About the Multi Value Property Window Pane

The Multi Value Property Window (MVPW) pane allows you to view and set values for multiple properties when you use one of the following designers:

- Entity Relationship Designer
- Task Designer
- Workflow Process Designer

For more information about how to use the MVPW pane, see *Siebel Business Process Framework: Task UI Guide* and *Siebel Business Process Framework: Workflow Guide* .

About the Bookmarks Window

The *Bookmarks window* is a window that allows you to navigate directly to an object that you use frequently. For more information, see *History Toolbar*.

Favorites in Web Tools

In Web Tools you can save objects you are configuring as a Favorite. A Favorite can be used to navigate back to the object from any location in Web Tools. This makes it easier for you to return to work in progress or to re-examine an object that has recently been of interest.

Saving an Object as a Favourite

To Save an Object as a Favorite

1. Search for the object you wish to save as a Favorite.

   Note: You can only select a single object at a time to be a Favorite. You can only save a list of objects as a Predefined Query

2. From the Applet Menu choose **Add To Favorites**.
3. A dialog will pop up. Only the *Name* field is editable.
4. Rename your Favorite to identify it when it appears in a list of other Favorites.

Using a Favorite

To Use a Favorite

1. In the Web Tools Application Menu click **Tools > Favorites**.
2. This will take you to the list of your Favorites. In the Favorites Column search for your Favorite by name.
3. Once you have located the record for the Favorite you wish to use, click the Drilldown in the Favorites column for that record.
4. You will be navigated back to your object in the current Workspace.

Note: Favorites are not Workspace aware. Clicking on a Favorite will attempt to open the object in the current Workspace.

- If the object does not exist in the current Workspace, you will not be navigated to the object.
- If the object does exist in the current Workspace, you will be navigated to it as it is in the current Workspace.
- If you have modified that object in a different Workspace, you will still see the object definition in the current Workspace so your changes in another Workspace will not be reflected.
- You must first open the Workspace in which the modified object you wish to view exists.

Deleting a Favorite

To Delete a Favorite

1. Navigate to the Web Tools Application Menu.
2. Click **Tools > Favorites** .
3. In the Favorites Column search for your Favorite by name.
4. Once you have located the record for the Favorite you wish to delete, click the trash can icon or choose Delete Record from the Applet Menu.

Predefined Queries in Web Tools

Just as in the customer facing application you can create and use Predefined Queries. This will save a query you have created for later use. If a Predefined Query is associated to a particular object type it will show in the Predefined Query drop down on the Application Menu Bar. If a Predefined Query is currently in effect, it will be displayed in the dropdown. When you select a Predefined Query from the dropdown list it will execute and return the records for the Applet.

Creating a Predefined Query

To Create a Predefined Query

1. For any object type in Web Tools execute a query.
2. When the record set is what you want, click **Tools > Queries > Save Query As** or use the **ALT + S** keyboard shortcut.
3. When the *Save Query As* dialog pops up, give the query a name in the *Query Name* field that will clearly indicate what the query is designed to do.
4. After entering the name for the query click the **OK**.
5. The Predefined Query dropdown in the Application Menu Bar will now display this query name.

Using a Predefined Query

Predefined Queries are tied to a specific View in Web Tools. That also means they are tied to a specific object type such as Business Component.

If the query you are looking for does not appear in the Predefined Query dropdown in the Application Menu Bar, check to see what object type you are querying.

To Use a Predefined Query

1. Navigate to the object type you wish to query such as Business Component.
2. Choose your query from the Predefined Query dropdown in the Application Menu Bar.
3. The query will execute and return objects in the current Workspace.

Refining a Predefined Query

You can refine a Predefined Query and update it, replacing the previous query with your refined query.

To Refine a Predefined Query

1. Navigate to the object type you wish to query such as Business Component.
2. Choose your query from the Predefined Query dropdown in the Application Menu Bar.
3. Once the query has completed click the Applet Menu and choose **Refine Query** or use the ALT + G keyboard shortcut.
4. Update the query conditions as necessary and execute the query.
5. With the adjusted query results as you need them displaying in the Applet, click **Tools > Queries > Save**.
6. Your query displayed in the Predefined Query dropdown will be updated with the new criteria.

Examining or Deleting a Predefined Query

You may wish to examine the syntax of a Predefined Query or delete it. To do this navigate to the *Predefined Queries View*, find your query, and examine or delete it.

To Examine or delete a Predefined Query

1. Navigate to the *Predefined Queries View* by clicking **Tools > Queries > Predefined Queries** in the Application Menu.
2. In the *Predefined Queries View* find the query you wish to examine or delete.
3. Once you have located the correct query, examine the *Query* field to see the syntax for the query.

    > **Note:** You can modify the syntax in this View, but it is recommended that you do not manually adjust the query syntax as this is error prone.

4. If you wish to delete this query either click the trash can icon with the query selected or choose **Delete Record** from the Applet Menu.

    > **Note:** If your intent is to delete a query and not examine the query you can choose **Tools > Queries > Delete Saved Query.** This brings up a list of Predefine Queries that are associated with the current object type. You can quickly choose a query from that list and delete it rather than going to the *Predefined Queries View* to locate and delete it.

Running Queries

This topic describes how to run a query in the Object List Editor. If you use the Query Menu, then Object List Editor searches for objects according to the conditions that you enter in one or more properties. You can use a simple query or a compound query. You can create, refine, or activate a query from the Query menu or from the List toolbar. You can also use shortcut keys instead of using the Query menu.

Using the Query Menu to Run a Query

This topic describes how to use the Query menu to run a query.

To run a query

1. In the Object Explorer, click the object type you want to locate.
2. Click Query (the magnifying glass icon) in the Object List Editor.

3. In the empty record that appears in the Object List Editor, enter the query and then press ENTER (the return key).

   The records that meet the query criteria that you entered are returned.

   For more information, see *Simple Query Operators* and *Compound Query Operators*.

   Note: You can also search using the Filter preceding the list of objects in the Object List Editor.

Using Shortcut Keys to Run a Query

You can use a shortcut key to run a query.

To use shortcut keys to run a query

1. In the Object Explorer, click the object type that you want to locate.
2. In the Object List Editor, enter `ALT+Q` in Web Tools.
3. In the Object List Editor, enter the query.

   For more information, see *Simple Query Operators* and *Compound Query Operators*.
4. Press the ENTER key.

   The query results are returned.

Removing Query Results from the Object List Editor

You can remove query results from the Object List Editor.

To remove query results from the Object List Editor

1. Click Query (the magnifying glass icon) in the Object List Editor.
2. Do not enter any values in the empty record that appears in the Object List Editor.
3. Press ENTER (the return key).

Simple Query Operators

The following table describes the operators that you can use to create a simple query. A *simple query* is a type of query that locates records according to one condition. A check mark in a property that can contain a Boolean value represents TRUE. The Changed property is an example of a property that contains a Boolean value. For more information about query operators and Siebel data types, see *Siebel Developer's Reference* .

Operator	Description
=	Equal to (equals operator).

Operator	Description
`<`	Less than (opening angle bracket).
`>`	Greater than (closing angle bracket).
`<>`	Not equal to (opening angle bracket and closing angle bracket).
`<=`	Less than or equal to (opening angle bracket and equals operator).
`>=`	Greater than or equal to (closing angle bracket and equals operator).
`*`	A wildcard. An asterisk (`*`) can represent any number of characters, including no characters.
`?`	A wildcard. A question mark (`?`) can represent any single character.
`IS NOT NULL`	Queries for a property that is not empty.
`IS NULL`	Queries for an empty property.
`LIKE`	Queries for a value that begins with the string that you enter.
`NOT LIKE`	Queries for a value that does not start with the string that you enter.
`" "`	Queries for a string that includes a special character. For example, enter `"MyQuery's Text"` to query for the `MyQuery’s Text` string. If your query text includes a special character, then use quotes to enclose the query.
`EXISTS()`	Queries for a value in a multi-value group.
`~`	Forces the text string to use the case that follows the tilde (`~`).

Compound Query Operators

A *compound query* is a type of query that locates records according to more than one condition. You can use parentheses to control the order that Siebel uses to execute a compound query. Siebel runs the query according to the expression that you enter. It starts with the inside parentheses and then proceeds to the next, similar to the order in which you read English.

The following table describes operators that you can use for a compound query. You can combine simple conditions and compound conditions in a single query. For more information, see *Siebel Developer's Reference* .

Operator	Description
AND	All the conditions that the AND operator connects must be true.
OR	At least one of the conditions that the OR operator connects must be true.
NOT	The condition that the NOT operator precedes must be false.

To use a compound query, do one of the following:

- Enter conditions in two or more properties.
- When you run the query, Siebel uses an AND operator between the conditions that you enter.
- Use OR, AND, and NOT operators in a single property.

Using Editors

The following table describes some of the editors that are available. These editors include a canvas and you can use them with the Palettes pane. A *canvas* is a background that appears in different designers. It includes a work area that allows you to move or rearrange objects. For example, the Task Designer allows you to move a Siebel Operation step from the Palettes pane to the canvas. The canvas also indicates whether or not you can edit an object.

Editor	Description
Entity Relationship Designer	Allows you to diagram the business entities that your configuration uses and represent relationships between these entities. For more information, see *Configuring Siebel Business Applications* .
Task Designer/Editor	Allows you to create a user interface that assists the user in completing a job task. Allows you to add steps and connectors to a task UI. For more information about task UI and using the Task Designer/ Editor, see *Siebel Business Process Framework: Task UI Guide* .
Workflow Process Designer	Allows you to define, manage, and enforce the business processes that your company uses. Allows you to add steps and connectors to a Workflow Process. For more information about Workflow Processes, using the Workflow Designer, and using the Workflow Process Simulator, see *Siebel Business Process Framework: Workflow Guide* .

Using a Script Editor

If you cannot use declarative configuration to meet your requirements, then you can use the Server Script Editor and the Browser Script Editor to create a script. You use these editors to add a script to a Siebel object. You can use the following types of script:

- Siebel VB
- Siebel eScript
- Browser Script

For information about declarative configuration, see *About Siebel Tools*. For information about scripting, see *Using Siebel Script Editors*.

Using the New Object Wizard

The *new object wizard* feature guides you through the steps that are involved in creating a new object. You can use the New Object Wizard to create new objects of the following type:

- **General object.** For example, a business component, table, view, or applet method menu item.
- **Applet object.** For example, a list applet, form applet, MVG Applet, or chart applet.
- **Task object.** For example, a task, task applet, task view, or transient business component.
- **EAI object.** For example, an integration object, a data access or Web service.

Note: The new object wizard is only partially supported in Web Tools - see step 2 in the following procedure.

To use the new object wizard

1. Click New Object Wizards (the magic wand icon) on the application banner in Web Tools.
2. Select the wizard that you want to create the new object.
   For more information on how to use the new object wizard to create a new General, Applet or Task object, see *Configuring Siebel Business Applications* .
   For more information on how to use the new object wizard to create a new Web service object in Siebel Tools, see the topic about Creating an Outbound REST Service Based on an Open API Compliant JSON File in *Siebel REST API Guide*
   Currently, the new object wizard is partially supported in Web Tools to create a new Web service object. For more information, see *Creating a New Web Service Object in Web Tools*.

Note: You can use the Edit Web Layout function, which is available only in Web Tools, to edit the web layout.

Creating a New Web Service Object in Web Tools

The Web Service Wizard lets you define Web services by importing WSDL or JSON files in to the Siebel CRM Repository. This provides a way to connect to external Web services without the need to manually define the external Web services (by creating, for example, the required integration objects and business services). The following procedure shows how to create a new Web service object, using the Web Service Wizard, in Web Tools.

To create a new Web service object in Web Tools

1. Click New Object Wizards (the magic wand icon) on the application banner.
   The following message appears if you are in a non editable Workspace: *Workspace is not editable. (SBL-DAT-60352).*

For more information about New Object Wizards, see *Using the New Object Wizard*.

2. Select Web Service from the list of wizards that appear and then click Start to start the Web Service Wizard.
3. Click Upload File to select the WSDL/JSON file, containing the Web service definition, you want to import.

   The name of the WSDL/JSON file appears in the File Name field. Alternatively, you can specify a URL from which to download the JSON/WSDL to import. To do this, click the radio button next to the URL. This will enable the field in which you can paste a URL.
4. Click Next to continue.

   The Web Service Wizard validates the WSDL/JSON file and displays a summary of the actions it will take. The wizard also generates some files for download, including the following:

   - The original input file (that you uploaded in step 3).
   - Intermediate transformation files, showing how the input was converted into a business service and integration objects.
   - Log files, which might help if the import fails or is successful but the result is not what you were expecting.
   - XML file for use in the runtime configuration.

     If you want to download these files, then you should do so now since these are temporary files which will be removed when you finish the wizard. If you forget to download the runtime configuration data file (the most important file), you can regenerate it from the Web Services Administration view.
5. Click Submit when you are ready to commit the changes.

   The XML runtime file will be generated.
6. Click Finish to close the wizard.

   The integration objects and business service will be placed in to the current Workspace and the XML runtime file will be available. If the file that you uploaded is not valid, then the wizard will show an error in the Validation Results window. In some failure cases, you can download additional log or intermediate files to get more information (these will only be available if the wizard was able to get far enough to generate them).

Using Web Templates

A *Web Template* is an applet in Web Tools that displays the HTML code of a Siebel Web Template in an editor and also lets you preview the object design.

To use the Web Templates applet

1. In the Object Explorer, select the Web Template object.

   To filter the templates, use the filter preceding the object list or run a query using the object's Menu (cogwheel icon) in the Object List Editor applet.
2. Drill down on the Web template that you want to view or modify.

   > **Note:** Web Tools displays the HTML code for Siebel Web Template in the Object List Editor. You open a Workspace to modify it - see *Opening Existing Workspaces*.

3. If the Web template is editable, you can edit the HTML code in the editor.

You can customize the editor with options such as auto-close tags, hints, display of line numbers or Object Design hierarchy, and indentation. The settings you select are automatically available whenever you access the editor.

4. Click Preview (the eye-shaped icon) to preview the object design.

   You can toggle between the editor and preview.
5. Click the Back button to exit the preview.

Using IDE in Web Tools

This topic describes how to use Integrated Development Environment (IDE) in Web Tools. It includes the following information:

- *Overview of Integrated Development Environment*
- *Using a Web Page Editor*
- *Using a View Web Template Editor*
- *Using an Applet Web Template Editor*

For more information about using these editors, see *Configuring Siebel Business Applications* .

Overview of Integrated Development Environment

In Web Tools, Integrated Development Environment (IDE) provides a selection of tools for you to create or modify a Siebel object. You can access IDE through the preview mode of View or Applet Web templates in the Object Explorer. IDE helps you configure new applets and views or modify existing ones. You can also preview your changes from within IDE for the desired application. To exit IDE and revert to Object Explorer, click Close on the IDE title bar. The following table describes the various elements of IDE.

Section	Type of Editor	Description
Title bar	Both	Shows the view or applet you selected for update. It also has the Close button to exit IDE.
Form Factor	Both	Helps you view the layout in three different form factors: Desktop, Tablet, and Mobile. This helps fine tune the layout as per your requirement and visualize the result. For example, placeholders that display in a row in Desktop will stack under each other in a column in Mobile due to restricted space.
Show More/Show Less	Applet Web Template Editor	This option appears next to the Form Factor buttons and toggles to display more or less fields and columns in applets. In grid-based form applets, you can fine-tune the fields that will appear by clicking the pin icon that appears on the control/column after you move it to the form. The pin is a toggle to enable/disable a (red) border around the control and its label. The appearance of the border indicates that the control will display only during the Show More mode of the applet.
Applications	Both	Shows the Siebel applications so that you can select the application that will reflect your changes. In the Applet Web Template Editor, you can select All Applications to make your changes available to all applications. You can also select a particular

Section	Type of Editor	Description
		application in which you want to use the applet. If you select a specific application, your changes will be available for the applet only in the selected application. In View Web Template Editor, use the preview option to view your changes within IDE. Before previewing, set the user property of the selected application. The preview displays the objects in context but without data. If you want to view the changes made in either editor with data, then log in to the actual application and submit the changes for delivery.
Library	View Web Template Editor	Lists preconfigured applets you can map to the selected view. To unmap, can click Delete (the Trash icon).
Web Template	Applet Web Template Editor	Displays all the web templates for a selected applet so that you can select to work on another web template from within the IDE itself. For each web template, this pane lists Web Controls that you can use to create a control or a column. In Controls/Columns, the pane lists preconfigured controls/columns and their labels for the applet. You can also filter on unmapped controls/columns to select a required object quickly. **Note:** In grid-based applets, the controls and their labels display separately under the Name (control) and Type (label) columns so you need to move them to the canvas one at a time. The difference between a control and its label is that the control has a border.
Canvas	Both	Contains placeholders to receive an object. Click the plus (+) icon to show additional unmapped placeholders. Click Delete (the Trash can icon) to unmap an object and reuse a placeholder. You can resize the canvas from both the Library and Property pane sides. **Note:** In the Applet Web Template, you can unmap individual web controls and controls/columns.
Properties	Both	Lists the properties of the currently selected object, web control, or control/column. If you click an empty placeholder, then this section will be blank. If you click outside a placeholder, on the canvas, then it will display the selected property of the view or applet.

Using a Web Page Editor

Web Tools supports Web Page Editor for both edit and preview of the code.

To use a web page editor

1. In the Object List Editor, locate the web page that you want to modify.
2. Click Preview (the eye-shaped icon) to preview the design in the same page as the object.

   Note: You can make changes to the web page in its record.

3. To save your changes to the web page or exit the preview, step off the record.

Using a View Web Template Editor

You sometimes have to modify an existing view to add a new applet or remove an existing one. You can do these tasks in View Web Template Editor, which is available in IDE. For more information about IDE, see *Overview of Integrated Development Environment*.

To use a View Web Template editor

1. Create or open a Workspace.
2. In the Object List Editor, click View and search for a view where you want to add a new applet.

   For example, search for Account List View.
3. In the View Web Template of the view, click Preview to view the IDE for View Web Template Editor.

   The IDE for the view appears. The Property pane in the IDE displays the property of the selected view web template similar to that in Object List Editor.
4. Select a form factor in which to view your changes. The default view is Desktop.
5. In the Library pane, search for an applet that you want to add to the view.

   For example, search for Account Contact List Applet.
6. Select the applet and, with the mouse button depressed, move the selected applet onto an unused placeholder and then release the mouse button. The Properties pane displays the property for the new applet.

   For example, move it to placeholder 14. If the placeholder is not visible, then click the plus sign on the canvas until you reach it or move to any unused placeholder.
7. To preview your changes in an application, select it from the Applications drop-down list (for example, select Siebel Universal Agent to preview changes in the Call Center application) and then proceed as follows:
   - If the application is configured as viewable, then click the arrow button next to the Application drop-down list to preview the changes from within Web Tools.

     Log in to the chosen application in the pop-up dialog box that opens in the View Web Template Editor.

     The preview shows your selected view and applet in the application.

     Note: You can view the selected application as per your login credentials but cannot make any change to it due to viewing it in the preview mode. To submit your changes, directly log in to the application, go to the Workspace Dashboard and then click Inspect.
   - If the application is not yet configured as viewable within Web Tools, then a pop-up message prompts you to set the URLApp application user property for the selected application.
     1. Click OK in the pop-up message.

        Web Tools automatically takes you to Application, Application User Props of Siebel Universal Agent in the Object Explorer.

        Note: You can click Cancel in the pop-up message to select another application or perform another task in the editor. However, the arrow button of the Applications drop-down list enables only after you enable the User Property of the application. At any time, you can click the Close button at the end of the title bar to exit the IDE.

ii. Create a new record in Application User Props and then enter the value as URLApp in the Name field.

iii. Enter the value as callcenter (or as appropriate for your customized application) in the Value field of the same record and repeat Step 3.

> **Note:** To know the value associated with the application, see the identifier that precedes the language in a typical Siebel URL such as `http(s)://<server>:<port>/siebel/app/<application name>/<language>` for `http://siebeldemo:16001/siebel/app/callcenter/enu`. The application name is also listed in the Application Interface Profile when you install Siebel. For more information about Siebel Application Interface Profile, see the *Siebel Installation Guide* .

iv. The View Web Template Editor for the required view web template displays and you can perform Step 7 to preview the application.

Using an Applet Web Template Editor

You can use Applet Web Template Editor to modify existing controls or add new ones. You can do so for any of the Web templates of the applet without exiting IDE to select the next Web template. You must access this editor through IDE. For more information about IDE, see *Overview of Integrated Development Environment*.

To use an Applet Web Template editor

1. Create or open a Workspace.
2. In the Object List Editor, click Applet, click Applet Web Template, and then search for the applet that you want to modify.

   For example, search for SIS Account Entry Applet.
3. Select the Web template that you want to modify and then click Preview to view the IDE for Applet Web Template Editor.
4. Select a form factor to preview your changes.
5. Select the application you want to modify from the Applications drop-down list.

   For example, select Siebel Universal Agent to view changes only in the Call Center application. If you select All Applications then the changes will reflect in all applications where this applet is in use.

   > **Note:** At any time, you can click the Close button at the end of the title bar to exit the IDE. To submit your changes, log in to the selected application, go to the Workspace Dashboard, and then click Inspect.
6. Select a Web control and, with the mouse button depressed, move the selected Web control to an unused placeholder and then release the mouse button.

   For example, select the Delete Web control.
7. In the Properties pane, modify the properties of the object as required. The Property pane displays the same properties as given in the Object List Editor.
8. Select a control and, with the mouse button depressed, move the selected control to an unused placeholder and then release the mouse button.

   For example, select Account Channel.

9. Do likewise for the label of the control.

    > **Note:** In list-based applets, when you move a control, its label automatically moves to the adjacent placeholder.

    For example, the label of Account Channel.

Using the Command Line

For more information about using the command line, see the following topics:

- *Batch Validation*
- *Using the Command Line to Run the Locale Management Utility*
- *Using the Command Line to Export Objects to an Archive*
- *Using the Command Line to Import an Archive*

5 Using Workspaces

Using Workspaces

This chapter describes Workspaces, the Workspace Dashboard, and how to use and administer Workspaces. It includes the following topics:

- *About Workspaces*
- *Workspace Dashboard*
- *Common Workspace Tasks*
- *Workspaces Administration*

About Workspaces

A *Workspace* provides a developer with a sandbox for editing and testing configuration changes before making them available to other users. This ensures isolation of changes made by one user from interfering with those made by another user in parallel.

The Workspace feature allows users to manage the configuration of repository artifacts by providing the following key capabilities:

- Users can make configuration changes to the repository without any impact on other users, thereby allowing multiple developers to work on the same repository objects in the Siebel database in parallel.
- Workspaces are organized in a hierarchical structure where the root Workspace is named MAIN. Child Workspaces, grandchild Workspaces, and so on are underneath MAIN.
- Within a given Workspace, only the incremental modifications (delta changes) from the parent Workspace are stored in the Workspace itself. For example: MAIN contains the entire repository but a child Workspace under MAIN will contain only the differences.

Workspace Types and Inheritance

This topic describes the different types of Workspaces. Workspaces inherit the object definitions from their ancestor Workspaces (parent, grandparent, and so on).

- **MAIN Workspace.** MAIN is the root Workspace and it contains the current master version of the Siebel Repository (which includes all object definitions). The configuration in MAIN typically mirrors the current production environment.
- **Integration Workspace.** Integration Workspaces (IWS) are created under MAIN or another IWS, and typically represent the configuration changes needed for an upcoming release or (for an IWS that is a child of another IWS) some feature that is being developed for a future release. The purpose of an IWS, as the name suggests, is to allow testing of features and bug fixes from different developers *integrated with each other*.
- **Developer Workspace.** Developer Workspaces are the only editable Workspaces, where developers can build and unit test repository changes. Individual developers will make their changes in a personal Workspace that they create under some IWS. The content of this Workspace might contain a bug fix or some other contribution

to a larger feature represented by the parent IWS. Developer Workspaces can be *inspected* by any user to validate functionality. When a Developer Workspace is delivered to its parent:

- The original record is still maintained (you still have the old version).
- A new record will be created with an increment to the version number.

Workspace Users

Workspace users are categorized into these groups:

- **Workspace Owner** is the creator of a Workspace.
- **Workspace Administrator** is the owner of the MAIN Workspace.

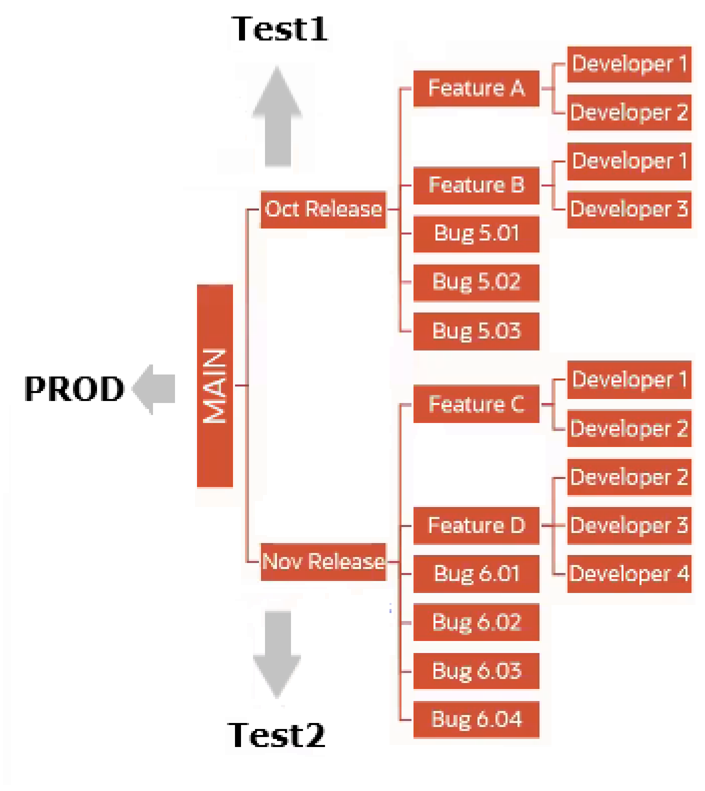
  The *Workspace Administrator* responsibility is required by any user who has to create or deliver Workspaces. Users with this responsibility can create integration and Developer Workspaces, deliver Workspaces to MAIN, rebase Workspaces, undo submit for delivery, flatten Workspaces, and so on.

  The *Composer Administrator* responsibility is also required to access the *Workspace Dashboard* in Web Tools so that users can perform developer and configuration operations. Users with this responsibility alone can only create Developer Workspaces, modify objects, checkpoint, and submit Workspaces for delivery.

  Developers are typically assigned only the Composer Administrator responsibility; they are not assigned the Workspace Administrator responsibility.

Basic Workspace Structure

The following image illustrates one example of the basic recommended structure for a Workspace.



As shown in this image, the main elements in a typical Workspace structure are as follows:

- There are releases being developed in parallel. For example: October and November releases.
- Each release has its own features. For example: October has Feature A and B, November has Feature C and D.
- Each release has bug fixes planned with Workspaces for each fix under the corresponding release IWS.
- Each feature has developers working on it with each developer having their part in a unique *Developer Workspace*. For example:
    - Developer 1 and 2 work on Feature A and Feature C.
    - Developer 1 and 3 work on Feature B.
    - Developer 2, 3, and 4 work on Feature D.
- As developers finish work on a bug or partial feature, QA – who are part of the development team – should help unit test changes in the Developer Workspace.
- The Developer Workspace should be delivered to the parent branch for integration testing only when both developer and QA are satisfied that it is operating properly when tested independently of other features and/or bug fixes. For example: When Developer 1 finishes their work on Feature A and the QA team is satisfied, the developer will deliver to the parent Integration Workspace (Feature A).
- After delivery to the parent Integration Workspace (IWS), the testing should focus on determining whether *this set of developer changes integrates well with other developer changes*. Since you have already unit tested it and confirmed the unit test by QA, all you are worried about in the IWS is whether the individual developer changes conflict with other developer changes or not.
- When an entire feature is ready, the feature is delivered to the release level. For example:
    - Feature A and B are delivered to the October Release level.
    - Feature C and D are delivered to the November Release level.

You can migrate the October release content to a UAT (User Acceptance Testing) environment and test it there while continuing to work on the October release and the November release in parallel.

- Test the release (User Acceptance Testing and Integration Testing).
- When the entire release is ready, deliver the Release to MAIN.
- Immediately migrate MAIN to Production.

  MAIN must always match your Production release.

Workspace Best Practice Guidelines

It is recommended that you adhere to the following best practice guidelines when dealing with Workspaces:

1. Do not limit the development environment just to developers. Adopt an agile software development model where the QA team is part of the development team, and where:
   - The QA team pre-validates changes in the Developer Workspace. The delivery of a feature to its parent is only allowed after pre-validation.
   - The Product Manager validates features in the development environment.

   This allows for the early detection of defects, leads to better quality features, and leads to more stable downstream environments. The agile model allows any design gaps that are identified to be fixed straight away.
2. Set up and plan for the following downstream test environments: User Acceptance Testing, Integration Testing, and Pre-Production Testing.
   - Populate the various test environments with the Integration Branches in the Design Time Repository (DR).
   - Migrate the changes in a given Integration Workspace (IWS) to downstream Runtime Repository (RR) environments as follows:
     - Populate downstream Test environments with the Integration Branches for a particular release. Once the release is approved for Production deployment, deliver the Integration Branches to the root MAIN branch.
     - Always update Production environments from the MAIN branch.
   - The Test environment reflects the next release so populate it as soon as the previous release has shipped and populate it repeatedly as changes are delivered. You can have more than one test environment.
3. When ready, deliver the Release to MAIN.
4. Immediately migrate MAIN to Production.

Workspace Limitations

Workspace limitations include the following:

- **Developer Workspace.** The maximum number of Developer Workspace versions that can be created under an Integration Workspace is 10,000. To avoid significant performance degradation, however, the recommendation is not to exceed 500 versions on a single Workspace.
- **Integration Workspace.** There is no limit on the number of Integration Workspaces that can be created for parallel development in a database. As the number of Workspaces increases, performance may start to

degrade. If this occurs, you can delete Integration Workspaces when they are no longer needed. For more information, see *Deleting Developer Workspaces*.

- **Workspace Hierarchy.** There is no specific limit on the depth of your Workspace hierarchy.

  Each Workspace layer, however, requires additional logic to determine the definition of an object. For example, consider a situation where you have changed Field_X in the Account Business Component in a Developer Workspace that is under a feature Integration Workspace (IWS), under a release IWS, under the MAIN Workspace. In this scenario, for the Account Business Component to be used at runtime to test the field change, the Object Manager must query for the following:

  - Field_X definition in the Developer Workspace, *plus*
  - any other changes made to the Account Business Component in the parent feature IWS, *plus*
  - any changes made in the release IWS, *plus*
  - the original definition in MAIN.

  Object Managers understand how to do this properly, but collecting all this data requires multiple queries, which reduces the runtime performance. There is no specific recommendations for the depth of the Workspace hierarchy, but a tree with too much depth may result in slow performance when performing configuration in a Developer Workspace.

  Note: This performance degradation will only apply to DR environments because there is no Workspace hierarchy in RR environments.

About Inspecting Objects in Workspaces

Integration Workspaces (including MAIN, which is a special example of an IWS) contain a compiled version of each object modified in that Workspace.

- MAIN contains a compiled version of every object.
- Each IWS contains a compiled version of each object modified and delivered from any child Workspace beneath it.

The compiled versions of the objects are stored in "S_RR*" tables. For example, compiled Applet definitions are stored in S_RR_APPLET. At runtime, the Application Object Manager reads the definitions of the objects from these tables.

Developer Workspaces, however, do not contain compiled versions of each object. Rather, they contain any changes to the raw repository definitions. To test these changes before delivering them to a parent Workspace, select the Inspect option from the *Workspace Dashboard*. This option will compile any changed objects in memory as they are needed, rather than read from the Runtime Repository table definitions.

How LOVs Work in Workspaces

How List of Values (LOV) or multilingual LOV (MLOV) data is created or updated in Siebel CRM has evolved relative to earlier releases. In particular, LOVs are now Workspace-enabled, allowing parallel development and sandboxing.

This topic summarizes some of the key changes in the LOV framework that were made starting in Siebel CRM 17.0 and continuing in subsequent Siebel CRM update releases. Changes described for LOVs also apply to MLOVs.

LOVs can be optionally created or updated in the following contexts:

- In the Design Repository by developers.
- In the Runtime Repository by administrators.

> **Note:** When you create an LOV in an Integration Workspace (IWS_A), the LOV is automatically created in every Integration Workspace whether existing or new (for example, in IWS_B, IWS_C and so on) but will be inactive in those Workspaces. To make the LOV available in IWS_B, you must deliver the current Workspace to its parent which activates the LOV in the parent, and so on up the delivery tree until you get to MAIN. The behavior is similar when rebasing a child Workspace – that is, any LOVs active in the parent will become active in the child Workspace. For more information about working with LOVs, see *Siebel Applications Administration Guide* .

LOV Changes for Design Repository

Because LOVs are Workspace-enabled, they are managed in the Design Repository (DR) environment in Workspaces, just like other repository objects, such as applets or business components. Unlike traditional repository objects, however, LOVs are stored at the Integration Workspace level. This allows one developer to create all the LOVs needed for a feature and the other developers to have them available in their sibling Workspaces, without the need for a rebase.

At the database level, all Integration Workspaces contain a full copy of all LOV records. This is done for two reasons: for performance and to avoid duplicate key errors during delivery.

When a developer creates a new LOV, that LOV will be active in that Workspace, with Workspace Inactive Flag = False (`WS_INACTIVE_FLG = N`). This record is propagated to all other Integration Workspaces as inactive, with Workspace Inactive Flag = True (`WS_INACTIVE_FLG = Y`). This ensures that no duplicate record can be created in another Workspace, while maintaining the sandboxing of LOVs to the Workspace in which they are created.

When the Integration Workspace is delivered to its parent, the copy of the LOV in the parent is updated to also be active, reflecting that it is now available in that parent and can be migrated to Runtime Repository (RR) environments by the Siebel Migration Application (along with any repository changes in that same Migration package).

When other Integration Workspaces are rebased, active LOVs are propagated into their child Integration Workspaces, making them available there as well.

The new columns in the S_LST_OF_VAL table that support the Workspace-enablement of LOVs are as follows:

- WS_ID (Workspace Id) stores the ROW_ID value of the corresponding Workspace record from the S_WORKSPACE table.
- WS_SRC_ID (Workspace Source Id) stores the ROW_ID value of the original LOV record (from the Workspace in which the record was created). When an LOV record is propagated to other Workspaces, the WS_SRC_ID value remains the same, ensuring that there is one logical record per LOV across all Workspaces. Delivery and rebase processes have no additional complexity as a result.

- WS_INACTIVE_FLG (Workspace Inactive Flag) is a Boolean column that stores the inactive flag (Y is inactive, N is active) for the LOV record in a given Workspace, as previously explained.

  WS_INACTIVE_FLG must be N for the LOV to show up in the List of Value Administration View for the current Workspace context. WS_INACTIVE_FLG itself is not visible in the user interface but rather is managed by the delivery and rebase processes.

  > **Note:** WS_INACTIVE_FLG is not to be confused with the existing ACTIVE_FLG column, which has always been part of the S_LST_OF_VAL table. ACTIVE_FLG maps to the Active field in the List of Value Administration View and allows an administrator to hide a row from end users at runtime by unchecking the Active checkbox.

> **Note:** As noted, while developers typically appear to make LOV changes in their Developer Workspaces, the actual LOV records are maintained at the parent Integration Workspace level. The same LOV can therefore also be changed directly in the Integration Workspace. It is possible to allow updates to only be made at the Integration Workspace level by setting the system preference `DisableSeedUpdOnDevWS = True`.

LOV Changes for Runtime Repository

While LOV data can be managed in the Design Repository (DR) environment using Workspaces and migrated to Runtime Repository (RR) environments in conjunction with repository objects in the same source Workspace, this data can also be managed in the RR environments themselves.

For example, a developer can create new LOVs in the DR environment for new or existing LOV types in order to perform testing and provide a base set of LOVs to migrate to the RR environment with a new or updated feature. At the same time, an administrator can add more values directly into the RR environment on an as-needed basis, without having to first put it into the DR environment and perform migration.

Runtime repository environments use only a MAIN Workspace (that is, there are no Integration or Developer Workspaces). MAIN supports multiple versions, which allows rollback to a last known stable version in the event of an issue with newly migrated objects. During a full repository migration from the DR to the RR, a full set of LOVs are copied into the target environment as version 0. As incremental migrations are performed, the MAIN version number increases by one, and any new or modified objects, including repository objects and LOVs, are tagged with the new version number. For this reason, there could be two or more physical records in the database representing the same logical record: one with an earlier version and others with later versions.

Each of the physical records (instances of the same logical record) for an LOV share the same WS_SRC_ID, but have different versions, allowing the Application Object Manager to select the correct instance at runtime. Each of these instances has an arbitrary ROW_ID value, just like any other newly created record. The WS_SRC_ID value is the same as the source DR record's WS_SRC_ID value, tying all instances and versions of an LOV logically together across all DR and RR environments.

Additionally, the following columns apply for LOV records in the Runtime Repository:

- WS_MIN_VER (Workspace Minimum Version) stores the minimum repository version in which this LOV record is valid. This value is 0 for any LOV record added through full repository migration. For any LOV record added or updated through incremental repository migration, the value represents the number of times incremental repository migration has occurred to that point, including the migration in which this LOV record migrated. A new record is created for an update.

  For example, where an LOV record was updated in the fifth incremental repository migration, WS_MIN_VER is set to 5 for the new record, which is valid as of MAIN version 5. (If you later roll back to an earlier version of

MAIN, then an older record can become valid again. At runtime, the Object Manager picks the record with the largest WS_MIN_VER value that is less than or equal to the version to which you rolled back.)

- WS_MAX_VER (Workspace Maximum Version) stores the maximum repository version below which this LOV record is valid. 10000 is the hardcoded maximum version and is the WS_MAX_VER value for any LOV record that has not been superseded through subsequent updates. Where incremental repository migration updated an existing LOV record, this value is reset for the original record to match the WS_MIN_VER value for the new version of the LOV record.

  For example, where an LOV record was updated in the fifth incremental repository migration, WS_MAX_VER is set to 5 for the prior version of this record, which is valid in MAIN version 4 but not in MAIN version 5. (If you later roll back to a prior version of MAIN, then such a prior version of the record can become valid, where the WS_MAX_VER value is higher than the repository version you are rolling back to.)

As noted, Siebel CRM enables users such as business administrators to use the Siebel application user interface to update LOV records directly in the Runtime Repository (in QA or production environments). As explained later in this topic, a conflict could occur as a result of an LOV record being updated directly in the Runtime Repository. A system preference is provided to help you manage such conflicts.

Managing Conflicts in LOV Record Versions During Migration

A conflict could occur as a result of an LOV record being updated directly in the runtime repository. If a record that originated in the Design Repository is updated in the runtime repository, is later changed in the Design Repository as well, and an incremental repository migration is performed, then the two versions of this record would conflict.

Because the record originated in the Design Repository, both records would have the same value for WS_SRC_ID, which stores the ROW_ID value of the LOV record in the source Workspace. WS_SRC_ID serves as a unique identifier across all environments, as described earlier. (A new record created in the Runtime Repository would not have a WS_SRC_ID value from the Design Repository, so would present no conflict to be resolved.)

To allow LOV records that originated in the Design Repository to be updated using multiple methods without conflicts during incremental repository migration, use the Seed Migration Priority system preference to designate which updates have priority. In the downstream or target environment (the Runtime Repository), set the system preference as follows:

- Seed Migration Priority = Target (the default) gives priority to the updates made in the Runtime Repository.
- Seed Migration Priority = Source gives priority to updates made in the design repository.

Additional LOV Changes

You can use EIM for bulk loading of LOV records in the Design Repository or in the runtime repository. For information about importing LOV data using EIM, see *Siebel Enterprise Integration Manager Administration Guide* .

Related Books

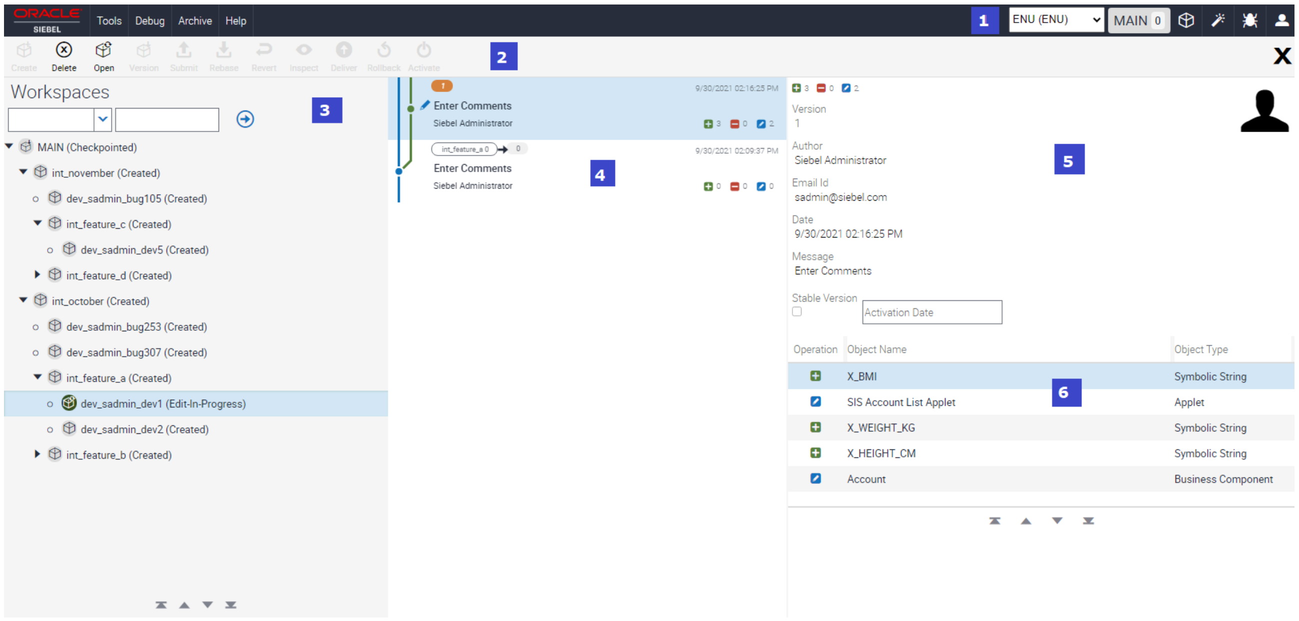
For more information about working with LOVs in the Siebel application user interface and about rolling back to a prior version of the Runtime Repository, see *Siebel Applications Administration Guide* .

For more information about performing repository migrations, see *Siebel Database Upgrade Guide* .

For information about importing LOVs using EIM, see *Siebel Enterprise Integration Manager Administration Guide* .

Workspace Dashboard

The Workspace Dashboard displays a list of all Workspaces that were created and are currently present in the database. From the Workspace Dashboard, you can perform various operations on Workspaces. The following image highlights the main elements of the Workspace Dashboard in Web Tools. A similar Workspace Dashboard is available in Siebel Tools.



MAIN (Checkpointed)
int_november (Created)
dev_sadmin_bug105 (Created)
int_feature_c (Created)
dev_sadmin_dev5 (Created)
int_feature_d (Created)
int_october (Created)
dev_sadmin_bug253 (Created)
dev_sadmin_bug307 (Created)
int_feature_a (Created)
dev_sadmin_dev1 (Edit-In-Progress)
dev_sadmin_dev2 (Created)
int_feature_b (Created)

Enter Comments
Siebel Administrator
9/30/2021 02:16:25 PM

int_feature_a 0
Enter Comments
Siebel Administrator
9/30/2021 02:09:37 PM

Version
1
Author
Siebel Administrator
Email Id
sadmin@siebel.com
Date
9/30/2021 02:16:25 PM
Message
Enter Comments
Stable Version
Activation Date

Operation	Object Name	Object Type
+	X_BMI	Symbolic String
✎	SIS Account List Applet	Applet
+	X_WEIGHT_KG	Symbolic String
+	X_HEIGHT_CM	Symbolic String
✎	Account	Business Component

As shown in this image, the main elements of the Workspace Dashboard are as follows:

1. **Application-Level Toolbar.** The items on the application-level toolbar related to Workspaces are:
   - **Workspace Name.** For example: `MAIN`.
   - **Workspace Version number.** For example: `MAIN 0`.
   - **Workspace (or cube) icon.** Click to open the Workspace Dashboard (if not already open). Click Close (the X icon) to exit or close the Workspace Dashboard.
2. **Workspace Toolbar.** Contains various options to configure and manage Workspaces. The following table describes the options available on the Workspace Toolbar. These options may be active or inactive, depending on the current Workspace user and the selected Workspace.

Button	Description
Create	Click this button to create a new Workspace. For more information, see *Creating New Workspaces*.
Delete	Select a Workspace and click this button to delete the selected Workspace. For more information, see *Deleting Developer Workspaces*.

Button	Description
Open	Click this button to open an existing Workspace.
Validate	Click this button to validate a single or multiple objects within an object type, or a Workspace. For more information, see *Validating Objects*.
Version	Click this button to create a version for the Workspace.
Submit	Click this button to change the status of the Workspace and make it ready for delivering the changes to the parent (MAIN) Workspace. For more information, see *Submitting Workspaces for Delivery*.
Rebase	Click this button to apply the changes that were made in the parent Workspace into the current Workspace. For more information, see *Rebasing Workspaces*.
Revert	Click this button to revert the changes that you made to the current version of that Workspace since the last checkpointed version. For more information, see *Reverting to Previous Workspace Versions*.
Inspect	Click this button to preview and test your changes in the Development Workspace (prior to delivering the changes to the parent Integration Workspace).
Deliver	Click this button to deliver the changes in the Workspace to the MAIN Workspace. This button is available only to the user who is also the owner of the MAIN Workspace. For more information, see *Delivering Workspaces*. **Note:** This option is available only to the user who is also the owner of the MAIN Workspace.
Rollback	Click this button to roll back the changes, in a production environment, to the last stable version of the runtime MAIN Workspace. This button is available only to the Admin user.
Activate	Click this button to activate a Workspace, in a production environment, to the latest stable version of a runtime MAIN Workspace. This button is available only to the Admin user.
Close (X icon)	Click this button to close the current Workspace.

3. **Workspace Explorer.** Lists all the available Workspaces in the database and their status. Depending on the Workspace that you select, different context menu options are available in the Workspace Dashboard.

   By default, the latest version of the root Workspace opens and this Workspace is set at the session. Also, the root Workspace MAIN is the only read-only Workspace.

4. **Workspace Version List Applet.** Shows a history for each version of the Workspace, including the version number, author, and any comments provided at the time that version was created.
5. **Workspace User Info Form Applet.** Shows the following information for the selected Workspace: version number, author, email Id of author, date and time the Workspace was modified, and message (if any).
6. **Workspace Objects Modified List Applet.** Lists the root-level objects that were modified in the selected version of the Workspace. This applet displays the object name, object type, and operation type performed on the modified objects.

Common Workspace Tasks

This topic describes how to perform various common Workspace tasks. These tasks are performed by all users who have the permissions to use Workspaces. This topic includes the following information:

- *Opening Existing Workspaces*
- *Editing Workspace-Enabled Repository Objects*
- *Creating New Workspaces*
- *Deleting Developer Workspaces*
- *Refreshing the Workspace Explorer*
- *Tracking Repository Object Changes in Workspaces*
- *Performing the Checkpoint Version Process*
- *Submitting Workspaces for Delivery*
- *Detecting Conflicts in Workspaces and Applying Resolutions*
- *Reverting to Previous Workspace Versions*
- *Rebasing Workspaces*
- *Canceling the Workspace Delivery Process*
- *Delivering Workspaces*
- *Comparing Workspace Versions*
- *Exporting Workspaces to an Archive*
- *Adding the Workspace Prefix System Preference*
- *Configuring Non-Workspace Objects*
- *Workspace Enabled Admin Data*
-
- *Using Workspaces for Seed Data*
- *Publishing Tables*

Opening Existing Workspaces

You must open a workpace to view it, but you can only edit Developer Workspaces.

- If you are the owner of the Developer Workspace, then you can edit (read and write to) that Workspace.

- If you are not the owner of the Developer Workspace, then you can only read (without editing) that Workspace.

Within your current session, you can only open one Workspace at any given time. Any configuration changes that you make are saved in that Workspace.

To open an existing Workspace

1. In the Workspace Explorer, expand the MAIN (parent) Workspace to display the complete list of all child Workspaces.
2. Select the Workspace that you want to open and then click Open on the Workspace Toolbar.

   The selected Workspace opens.

Editing Workspace-Enabled Repository Objects

Previous to Siebel CRM 17.0, in order to modify an object, it was necessary to lock the object or the project in which the object resided. In Siebel CRM 17.0 and later, you can edit Workspace-enabled repository objects directly without locking any project. Note that only Workspace-enabled repository objects can be edited inside Workspaces. By default, all Workspace-enabled objects are locked at the database level, so you are not required to lock projects when you edit the objects.

Workspace-enabled repository objects can be edited when the user opens a Developer Workspace.

Note: For objects that are not Workspace-enabled (Tables and other schema objects), you must lock the project or object before editing it. For more information, see *Configuring Non-Workspace Objects*.

To edit repository objects

1. In the *Workspace Dashboard*, create a new Workspace or open an existing one.
2. Edit the repository objects as required.

   For example, create new object records using the new object wizard and save the changes. For more information, see *Using the New Object Wizard*
3. Return to the Workspace Dashboard and check the changes that you made by reviewing the information in the Workspace Version, Workspace User Info Form, and Workspace Object Modified List applets.
4. Perform the Rebase process.
5. Perform the Deliver process to deliver the changes to the MAIN Workspace.
6. Confirm that the Deliver process has successfully completed. If not, continue to Step 10.
7. Navigate to the Conflicts Resolution view.

   By default, the values in the From Version Value list are committed to the target Workspace's To Version Value list during the Rebase process.
8. (Optional) Override the values in the To Version Value list and select the Override option.
9. Save the record.
10. Resolve all conflicts as needed.
11. Perform the Checkpoint process on the rebased Workspace.
12. Click Finish.
13. Click Deliver and then click Start to merge the changes to the mainline.
14. Click Finish to complete delivery to the MAIN Workspace.

Creating New Workspaces

This topic describes how to create new Workspaces. When creating a new Workspace, note the following:

- By default, newly recreated Workspace are always branched out from the latest version of the MAIN Workspace; therefore, the Parent Workspace field is populated with the value MAIN and the Parent Workspace Version field is populated with the latest check-pointed version of the MAIN Workspace.
- Workspace names can only contain lower case alphabetic, numeric, hyphen, full stop or period, and underscore characters. Also, Workspace names must start with the value that is set for Workspace Prefix in the System Preferences, appended by `_<login userid>_`. For example, if you log in using user name `admin1`, the Workspace name must be `dev_admin1_<myWorkspace>` where `dev` is the value set for the Workspace Prefix system preference. For more information on how to add Workspace Prefix in system preferences, see *Adding the Workspace Prefix System Preference*.

To create a new Workspace

1. In the *Workspace Dashboard*, click Create on the Workspace Toolbar.

   Note: Click the Workspace (or cube) icon on the application banner to open the Workspace Dashboard, if not already open, in Web Tools.

2. In the Create Workspace dialog that opens, enter the Workspace name in the Name field and a comment in the Comment field.
3. Click Create Workspace.

   In the Workspace Explorer, the newly created Workspace with the name that you entered is listed under the MAIN Workspace.

Deleting Developer Workspaces

This topic describes how to delete Developer Workspaces. Note the following about deleting Developer Workspaces:

- Users with the Workspace Administrator responsibility can delete Developer Workspaces.
- Developer Workspaces that are in a status of Submitted for Delivery or Delivered cannot be deleted from the repository.
- You can delete Developer Workspaces that you own but you cannot delete Workspaces or parent Workspaces that are owned by others.
- You cannot delete any parent objects or child-level objects if any of them are Workspace-enabled objects.
- Only the new parent objects that were created in one Workspace version can be deleted in that version.

To delete a Workspace

Note: Exercise caution when deleting a Workspace since this procedure removes a Workspace not just from the Workspace Explorer, but deletes all undelivered changes made in that Workspace.

1. In the Workspace Explorer (in the *Workspace Dashboard*), select the Workspace that you want to delete.
2. Click Delete on the Workspace Toolbar.

3. Click Delete Workspace in the confirmation dialog that appears.

   The selected Workspace is removed from the Workspace Explorer.

Deleting Developer Workspaces Using the siebdevcli Utility or Siebel Web Tools UI

Siebel developers with the Workspace Administrator Responsibility can delete Developer Workspaces after they have been delivered using the siebdevcli utility or by selecting and deleting them in the Siebel Web Tools user interface. This is especially useful for Developer Workspaces created directly under the MAIN Workspace. These types of Workspaces are typically created to deliver a hot fix to Production, and after being delivered, could not previously be deleted.

> **Note:** The command line switches `DeleteDeliveredDevWS` and `DeleteWorkspace` are case sensitive.

This first use will delete all delivered Development Workspaces under a specified parent Integration Workspace.

```
siebdevcli /u <UserName> /p <Password> /d <DataSourceName> /c <ConfigFile> [/l
<Language>] /DeleteDeliveredDevWS <integ_Workspace_name>
```

For example, this command will delete all those Developer Workspaces which are in delivered state under `int_test`.

Windows:

```
siebdevcli.exe /u ******* /p ******* /d "ServerDataSrc" /c
"C:\2474\bin\enu\tools.cfg" /l ENU /DeleteDeliveredDevWS int_test
```

Linux:

```
./siebdevcli /u ***** /p ***** /d "siebel_DSN" /c
"/export/home/sblqa1/2024_07C004/ses/siebsrvr/bin/export.cfg" /l ENU /DeleteDeliveredDevWS int_test
```

The following will delete all the delivered Development Workspaces from a particular version of the parent Integration Workspace by following the Integration Workspace name by the version number.

```
siebdevcli /u <UserName> /p <Password> /d <DataSourceName> /c <ConfigFile> [/l
<Language>] /DeleteDeliveredDevWS <integ_Workspace_name> <Version>
```

This example shows deleting all the delivered Developer Workspaces from the parent Integration Workspace `int_test` for version `4` of that Workspace. All other versions will be unaffected.

Windows:

```
siebdevcli.exe /u ***** /p ***** /d "ServerDataSrc" /c
"C:\2474\bin\enu\tools.cfg" /l ENU /DeleteDeliveredDevWS int_test 4
```

Linux:

```
./siebdevcli /u ***** /p ***** /d "siebel_DSN" /c
"/export/home/sblqa1/2024_07C004/ses/siebsrvr/bin/export.cfg" /l ENU /DeleteDeliveredDevWS int_test 4
```

You can also delete a single, delivered Developer Workspace by specifying it by name. Since Workspace names are unique across all Workspaces, you do not need to qualify it with its parent Integration Workspace. This uses the `DeleteWorkspace` switch.

```
siebdevcli /u <UserName> /p <Password> /d <DataSourceName> /c <ConfigFile> [/l
<Language>] /DeleteWorkspace <Developer_Workspace_Name>
```

This example shows how to delete a specific, delivered Developer Workspace.

Windows:

```
siebdevcli.exe /u ***** /p ***** /d "ServerDataSrc" /c
```

```
    "C:\2474\bin\enu\tools.cfg" /l ENU /DeleteWorkspace dev_ccheng_01
```

Linux:

```
    ./siebdevcli /u ***** /p ***** /d "siebel_DSN" /c
    "/export/home/sblqa1/2024_07C004/ses/siebsrvr/bin/export.cfg" /l ENU /DeleteWorkspace dev_sadmin_ccc1
```

In this example, you can specify a list of Developer Workspaces in a file. This is useful for deleting specific Workspaces when you do not want to delete all of them. This does not have to be done by a user with the Workspace Administrator Responsibility but can be done by the owner of the Workspaces just as they would in the Web Tools user interface.

Note: This command can delete Workspaces in the status of **Delivered, Created, or Edit-in-Progress**.

```
    siebdevcli /u <UserName> /p <Password> /d <DataSourceName> /c <ConfigFile> [/l
    <Language>] /DeleteWorkspaces <filename.txt>
```

Windows:

```
    siebdevcli.exe /u ***** /p ***** /d "ServerDataSrc" /c
     "C:\2484\bin\enu\tools.cfg" /l ENU /DeleteWorkspaces C:\fs\deletews.txt
```

Linux:

```
    ./siebdevcli /u ***** /p ***** /d "siebel_DSN" /c
     "/export/home/sblqa1/2024_08C004/ses/siebsrvr/bin/export8c4.cfg" /l ENU /DeleteWorkspaces deletews.txt
```

Note: If you try to delete a Workspace which is in **Submitted for Delivery** status by providing the Workspace name in a file, the Workspace will not be deleted. The `siebdevcli.log` will contain this error message:Workspace Name `"dev_sadmin_test"` cannot be deleted.

Deleting delivered Developer Workspaces is not limited to the siebdevcli utility. Developers can also delete them directly in Siebel Web Tools. To do this do the following:

1. Select the Workspace.
2. Click the Delete button.

Refreshing the Workspace Explorer Pane

When you open the Workspace Explorer, it lists all available Workspaces that are in the database. After that, if other users use other sessions to modify their Workspaces in the same database, such as adding new Workspaces or deleting the current ones, you must refresh your Workspace Explorer to display these changes.

To refresh the Workspace Explorer

1. In the Workspace Explorer (in the *Workspace Dashboard*), select and right-click the root (MAIN) Workspace.
2. Click Refresh/Reload current page.

   The Workspace Explorer is refreshed, displaying the latest changes for all Workspaces in the database.

Tracking Repository Object Changes in Workspaces

The Workspace Objects Modified List applet lists all the objects that were modified in the selected version of the Workspace, and includes the following information: Object Name, Object Type, and Operation Type performed on the modified objects.

To track repository object changes in a Workspace

1. In the Workspace Explorer (in the *Workspace Dashboard*), select the Workspace you want to track.
2. Select a version of the Workspace in the Workspace Version applet.
3. Review the changes in the Workspace Objects Modified List applet.

   The applet shows the following information: the object name, type, and operation that was performed for each modified object. The following table describes the values that can appear in the Operation column.

Option	Description
Update	Indicates an operation of updating a parent object record. Also indicates an operation of inserting, updating, or enabling the Inactive option for a child record.
Insert	Indicates an operation of inserting a new parent object record.
Delete	Indicates an operation of enabling the Inactive option for a parent object record.

4. To see only the changes in the repository, then open a version of the Workspace.

   When you open a particular version of a Workspace, all versions preceding the selected version are also opened. For example, if the Workspace `dev_sadmin_demo` has 10 versions and you open the fifth version, then you can view the repository changes of version one through version five.

Note: You cannot delete any existing record in Workspace-enabled objects; but the new objects that were created in the latest Workspace version can be deleted in that version.

Performing the Checkpoint Version Process

Note: This task applies only to Siebel Tools.

The checkpoint operation commits the changes that you made to the current Workspace version of the selected Workspace and sets that version to be non-editable. After checkpointing the Workspace, you are no longer able to make object changes to the current version of the Workspace. All subsequent object changes will be tracked under the next version of the Workspace.

To perform a checkpoint Workspace version process (Siebel Tools)

1. In the Workspace Dashboard, select the Checkpoint option from the Workspace menu.
2. Enter comments in the Enter Comment dialog box that appears. These comments will appear in the Comment column of the Workspace Versions applet.
3. Click OK (a message appears saying the checkpoint process is completed successfully) and then OK again.

   After a Workspace version is check pointed, its status changes to Checkpointed and the Submit for Delivery option for that Workspace is enabled in the Workspace menu.

Submitting Workspaces for Delivery

After you perform the checkpoint process on the latest Workspace version, you can use the Submit option on the Workspace Toolbar to deliver your Workspace changes to the MAIN Workspace. After the changes are submitted for delivery, the Workspace is read-only and no further configuration changes can be made.

> **Note:** It is possible to add governance and approval steps for delivering Workspaces using the Siebel Approval Manager product. For more information, see *Siebel Approval Manager Guide* .

> **Note:** When you select the Submit option, the system identifies whether the rebase process is required with the current modifications. If the merge process is non-trivial, then an error message appears reminding you to run the rebase process before the system converts the Workspace to read-only mode. When there is a trivial merge, the system changes the Workspace status to Submitted and converts the Workspace to read-only mode.

To submit a Workspace for delivery

1. In the *Workspace Dashboard*, click Submit on the Workspace Toolbar.
2. Click Yes on the confirmation message that appears to start the submit process.

   The Workspace status changes to Submitted. The Workspace is now read-only and you cannot perform any more operations on the objects in the Object Explorer.

   Alternatively, click No to cancel the submission so that you can continue to make changes to the objects.
3. If there is a non-trivial merge issue, then a message appears prompting you to perform the rebase process, after which you can submit the Workspace for delivery again.

   For more information, see *Rebasing Workspaces*.

Detecting Conflicts in Workspaces and Applying Resolutions

In a development environment where multiple users work in parallel making configuration changes concurrently in their own private Workspaces, conflicts might arise when the same objects and attributes are modified by different users in their own private Workspaces, and then the changes from these users are delivered to the MAIN branch. These situations often lead to non-trivial merges because of the conflicting changes between different users.

For Workspaces, the system detects the conflicts prior to Workspace delivery. If conflicts or errors are found, the system displays the issues along with the default resolutions. You must review the conflicts and resolve the issues.

In other words, during the rebase Workspace process, the system enables you to:

- Identify the conflicts by viewing the conflict flags on the attributes. Notice that some conflicts must be explicitly resolved before you can submit the Workspace for delivery again.
- Identify the errors by viewing the status message of the objects or attributes.
- View the status message that describes the error or conflict in details for each object or attribute.

If the system cannot resolve the conflict or the delivery process encounters an error while merging the changes, an error message appears. Although an attempt is made on all objects, the overall delivery process fails even if a single error is encountered.

Note: Overriding the default resolutions is only permitted while the status of rebase is Pending Resolution. After the status changes to Finished, modifications in the Merge Conflicts dialog box will not be applied.

Tip: To assist with further analysis on errors encountered during the rebase, see the log `WS_REBASE_<wsname><rebase attempt number>.log`, located in the standard Siebel log folder. For example, `...\ses\siebsrvr\log`.

Conflicts and errors can be broadly categorized as follows: Attribute-Level and Object-Level (for example: name conflicts, object inactive conflicts, and index violation errors).

Attribute-Level Conflicts and Errors

The default resolutions for the attribute conflicts can be overridden by selecting the Override option. Select the Override option with caution because you might overwrite other users' changes that were already checked into the MAIN Workspace. If you select the Override option by mistake, then you can clear it to resolve the issue.

Errors can happen on attributes when the merge process is unable to set the correct value. Common errors are when there are inconsistencies in the repository and a foreign key is missing.

You can analyze the error further by viewing the appropriate log files.

Name Conflicts

A *name* conflict occurs when there are two different objects of the same type that have the same combination of name and parent ID. A potential violation of user key is detected as a result of the rebase operation, and the merge process selects a default resolution of renaming the object that belongs to the child Workspace by appending the string -[Rebase]. Both object and name attribute are marked with a conflict flag in the Merge Conflicts dialog box.

Here is an example of how a name conflict can occur:

- Suppose that both User1 and User2 are working on changes for an upcoming release being developed in the IWS `int_January`. Each user creates a Developer Workspace under `int_January\0` as follows: `dev_user1_Workspace` and `dev_user2_Workspace`.
- User1 creates a new business component named `Widget` and delivers that change to the parent IWS, creating a new version `int_January\1`.
- User2, who is unaware of User1's changes, also creates a new business component with the same name (`Widget`), tests it, and attempts to deliver it to `int_January`.
- User2's changes cannot be delivered because this is a non-trivial merge and it requires a rebase.

- During rebase, the merge process detects that there is a name conflict so it renames the object in `dev_user2_Workspace` to `Widget-[Rebase]`, and shows this as a conflict in the Merge Conflicts dialog box.

If a name conflict is not resolved, then subsequent delivery fails by identifying the system-generated name. This conflict cannot be resolved by selecting the Override option, but you must finish the rebase process and resolve it in either one of the two methods described in this example:

To resolve this conflict, it should be determined whether the users intended to create two different objects or the same object.

- **Case1:** The users were creating two logically different objects but happened to choose the same name. In this scenario, the second user must rename the duplicate object to a different unique name (such as `Super Widget`).
- **Case2:** If the users were creating the same logical object, the second user could revert to a previous version containing the `dev_user2_Workspace` Widget object and export it to an archive (SIF) file, then import it into the rebased version, using the archive capabilities to select which attributes to keep.

Avoid Rebase and Naming Conflicts for Workflow Processes

Previous to the Workspace Enablement of Workflow Processes and Tasks, only one version of a Workflow Process could be "Completed" (or active) at a given time. When a change was required to a Workflow Process, the developer would create a new version – which would become "In Progress" and eventually would replace the older version once it was fully tested. At that point, the older version would become "Obsolete" and the "In Progress" version would switch to "Completed" – representing that it is the current version.

In a Siebel Database Upgrade or Incremental Repository Merge (IRM) scenario, coming from a pre-2017 version of Siebel CRM, it is possible that some Workflow Processes have Obsolete, In Progress, and Completed versions, with the In Progress versions representing work that is not yet complete. During the (IRM/Database Upgrade) repository merge process, the logic assumes that the *latest completed version* should become the active version in the new Workspace model, which would ultimately end up in MAIN\0 in the upgraded Siebel Repository. Because only one version of an object is allowed in a given Workspace/Version combination, the other versions are renamed to include their version number, allowing them to be kept moving forward.

For example, consider an "ABCD Workflow Process" that has three obsolete versions, one completed version, and two in-progress versions in the pre-Siebel CRM 2017 database. The following table shows what will happen to the names of these Workflow Processes during the IRM/Upgrade merge process.

Old Workflow Process Name	Old Workflow Process Version	Old Workflow Process Status	New Workflow Process Name After Upgrade	Inactive Flag
ABCD	1	Obsolete	ABCD: 1	Y
ABCD	2	Obsolete	ABCD: 2	Y
ABCD	3	Obsolete	ABCD: 3	Y
ABCD	4	Completed	ABCD	N
ABCD	5	In Progress	ABCD: 5	Y
ABCD	6	In Progress	ABCD: 6	Y

As shown in this table, the merge is able to keep all versions of the Workflow Process, regardless of status, and only the one that was active in the previous environment (with status of "Completed") remains active in the upgraded repository.

> **Note:** If there are multiple completed versions, then the latest completed version will be active.

Now consider a situation where the developer wants to resume work on version 6 of the Workflow Process having previously worked on version 6 in the old database before the upgrade put a hold on that work. The logical path to do this would be to create a workspace, remove the inactive flag, make the required changes, test, and deliver the Workflow Process. The goal, however, is to deliver the Workflow Process as "ABCD" (since it is intended to replace the existing version). The developer cannot simply rename the Workflow Process because there is already a Workflow Process with that name.

To address this issue, the developer should do the following:

1. In the DEV Workspace before attempting to rebase, rename the old version to "ABCD-OLD" (for example) and mark it as Inactive.
2. Rename the new, replacement workflow to "ABCD" (in this example).

Doing this ensures everything will be successfully delivered, and subsequently migrated, upon rebase or delivery.

Open Inactive Conflicts

An *object inactive* conflict occurs when the target/child Workspace being rebased changes an object that is inactivated in the source/parent Workspace. This type of conflict will occur if any object above the modified object in the repository hierarchy is inactivated in the parent Workspace. For example, if a user changes a Business Component field and the containing Business Component itself was inactivated in the parent, then this conflict will arise.

You can either ignore this conflict or resolve this conflict by checking the Override option for the Inactive attribute.

For example:

- Both User1 and User2 are working on changes for an upcoming release being developed in the IWS `int_January`. Each user creates a Developer Workspace under `int_January\0` as `dev_user1_Workspace` and `dev_user2_Workspace`.
- User1 inactivates the Contact Business Component and delivers this change to the parent IWS, creating a new version `int_January1`.
- User2 modifies the Account Status Field by making it force active and then tries to deliver the change to `int_January`, but the system enforces a rebase.
- On rebase, the merge process detects that the changes on the Account Status Field are no longer applied as the parent Contact Business Component is inactivated. Hence a conflict is logged against the Contact Business Component and the corresponding inactive attribute.

To resolve the conflict, select the Override option for the inactive attribute on the contact Business Component.

About Overriding Name Conflicts for Tasks

For Task UI workflows, resolving (or overriding) name conflicts is slightly different. Where there is a name conflict and the user chooses to override the conflict, then the parent record is suffixed with `-old:0` and the record status changes to inactive. For example:

- Suppose that User1 and User2 are working on changes for an upcoming release being developed in the IWS `int_January`. Each user creates a Developer Workspace under `int_January\0` as follows: `dev_user1_Workspace` and `dev_user2_Workspace`.

- User1 creates a new task named `Create Account` and delivers that change to the parent IWS, creating a new version `int_January\1`.
- User2, who is unaware of User1's changes, also creates a new task with the same name (`Create Account`), tests it, and attempts to deliver it to `int_January`.
- User2's changes cannot be delivered because this is a non-trivial merge and it requires a rebase.
- During rebase, the merge process by default detects that there is a name conflict so it renames the object in `dev_user2_Workspace` to `Create Account-[Rebase]`.

Suppose User2 wants to retain the Create Account record they created. In such a case, User2 can select the Override option for the attribute (click Workspace menu, select Merge Reports, and go to the Attribute Differences-Critical Conflicts section – for more information, see *Viewing Information About Merge Conflicts and Resolutions*). As part of the override in this case:

- User1's record is renamed to `Create Account-old:0`.
- User1's record status changes to inactive.
- User2's record remains the same.

Index Violation Errors

An index violation error conflict occurs when there are two different objects of the same type having the same combination of contents. A potential violation of the index is detected as a result of the rebase and the merge process cannot select any resolution. Hence the system displays an error on the object and fails the rebase process.

You can fix this error either by manually modifying the fields involved in the index or by performing the RevertObject operation on the object (on a single version or the entire Workspace). This fix ensures that there is no longer a violation and then users can run the rebase process again.

Deleting Unique Indexes

If Oracle adds a new index (for example, S_EVT_ACT_CAL_Index2 which uses the PAR_REPTEVT_MST_ID column) with the same columns as an existing custom index (for example, S_EVT_ACT_CAL_Index1 which has already been created and uses the same PAR_REPTEVT_MST_ID column), then errors will be encountered when running the PostInstallDBSetup utility and RepositoryUpgrade utility. To resolve this issue, do the following:

1. Manually delete or inactivate the custom index object under (in this example) S_EVT_ACT_CAL in the Siebel Repository. For example:

    a. Locate the index object S_EVT_ACT_CAL_Index1 under the S_EVT_ACT_CAL table.
    b. Delete the index object S_EVT_ACT_CAL_Index1.

    Note that it is not sufficient to delete the index object just at the database level, you must delete the index object from the repository as well.
2. Run the PostInstallDBSetup utility again.

    For more information about the Post Installation Database Update process and running the PostInstallDBSetup utility, see *Siebel Database Upgrade Guide* .
3. If there are Test, QA, or Production environments that also have the same index (S_EVT_ACT_CAL_Index1 in this example), then run the RepositoryUpgrade utility.

    For more information about the RepositoryUpgrade utility and how to run the utility, see *Siebel Database Upgrade Guide* .

Viewing Information About Merge Conflicts and Resolutions

Note: This task applies only to Siebel Tools.

The following procedure shows how to view information about merge conflicts and resolutions.

To view information about merge conflicts and resolutions (Siebel Tools)

1. In the Workspace Dashboard, select Merge Reports from the Workspace menu.

   The Merged Workspaces applet opens with the following sections:

   - **Merged Workspaces.** This section shows the following information for the merge process: Status of merge, Resultant Version, Base Version, From Version, and To Version.
   - **Object Differences.** This section shows the following information for the merge process: Object Name, Object Type, Top-level parent name and type, and Status of the selected object after the merge process is completed.
   - **Attribute Differences - Critical Conflicts.** This section shows the attribute information including the Workspace Resolution, Base Version Value, To and From Version values, Conflict, Override, and so on.
2. (Optional) Select the Override flag to override the default Workspace resolution.
3. Click Finish to complete the merge process.

   Alternatively, click Cancel to cancel the merge process.

Reverting to Previous Workspace Versions

When the changes that were made since the last checkpointed version of the Workspace cause issues, you can revert or roll back to any previously checkpointed version. You can also revert or roll back a checkpointed version to the previous checkpointed version.

- If you perform the revert process on a Workspace with the status set to Edit-In-Progress or Checkpointed, then all changes that you made in the latest version are lost and the Workspace reverts back to the last checkpointed version.
- If you perform the revert process on the first version of the Workspace, then all changes that you made in the first version are lost and the status of the Workspace reverts back to Created.

To revert to a previous version of a Workspace

1. In the *Workspace Dashboard*, click Revert on the Workspace Toolbar.
2. Click Yes on the Revert Confirmation message that appears to revert all changes back to the last checkpointed version.

   All changes that were made in the latest version will be lost. Alternatively, click No to cancel the revert operation.

Rebasing Workspaces

Rebasing (merging) a Workspace is applying the changes that were made in the parent Workspace (as a result of deliveries by other Workspaces that have branched off the same parent) into the current Workspace.

Only the owner of the Workspace can perform the rebase process for that Workspace.

You can perform the rebase process only after you perform the checkpoint operation on at least the first version of the Workspace. If you attempt to perform the rebase process for a Workspace that has no changes in its parent Workspace, then there is nothing to rebase and a message similar to the following will be returned: *The rebase failed with error: No changes found to rebase/deliver/checkpoint.*

To rebase a Workspace

1. In the *Workspace Dashboard*, click Rebase on the Workspace Toolbar.

   The Rebase Workspace dialog that opens contains the following fields: From Workspace, To Workspace, Merge Status, and a Start Rebase button.
2. Click Start Rebase to start the rebase process.

   A rebase process bar appears in the Merge Status section.
3. Confirm that (or wait until) the rebase process bar shows the process is completed.

   When the rebase process completes successfully, a `Rebase Completed` message appears in the Merge Status section. The status of the Workspace is updated to Rebase-in-Progress and the version of the Workspace is incremented by 1.
4. If there is any conflict, a Resolve Conflicts button appears in the Rebase Workspace dialog, which you must click to resolve the conflicts.

   Note: If you open a Workspace that is in a Rebase-in-Progress state, you will not be able to edit the Workspace-enabled objects. In this state, users must select Merge Reports from the Workspace menu to view and resolve any conflicts as a result of the rebase. If the default conflict resolution taken by the rebase is acceptable, then click Finish to publish any rebased objects into the rebased Integration Workspace. For more information, see *Detecting Conflicts in Workspaces and Applying Resolutions*.

5. In the *Workspace Dashboard*, click Finish on the Merged Workspaces applet to complete the merge process.

   Alternatively, click Cancel to cancel the merge process. The Workspace will be reverted to the last checkpointed version and its status is set to Checkpointed.

   Note: If you are the owner of the Workspace, you can set the Override option and use the Finish or Cancel buttons in the Merged Workspaces applet for that Workspace.

6. After you click Finish, enter comments in the Enter Comment dialog and then click OK.

   A message appears confirming the completion of the merge process.

   Note: If you close the Merged Workspaces applet without clicking Finish while the Workspace status is Rebase-In-Progress, then you cannot do any object changes until you resolve the conflicts by clicking Finish or clicking Cancel to cancel the merge process. You can reopen the Merged Workspaces applet by selecting Merge Reports from the Workspace menu.

7. Click OK on the confirmation message that appears.

   After the Rebase process in completed successfully:

   - The status of the Workspace is updated to Checkpointed. In My Workspaces, the version of the Workspace is incremented by one and the comments that you previously entered are also displayed.
   - Workspace resolution is set to the From Version value by default. When you click the Finish button, the value of From Version is saved in your Workspace. If you select the Override option, then the Workspace resolution changes to To Version and the value of To Version is saved in your Workspace when you finish the rebase process.

   > **Note:** While checking the Override flag, keep in mind that you are overwriting other users' changes that are already checked into the MAIN Workspace. Therefore, use this option carefully. For details on various errors and conflict scenarios during the deliver and rebase processes, see *Detecting Conflicts in Workspaces and Applying Resolutions*.

Canceling the Workspace Delivery Process

Before delivering a Workspace, Workspace Administrators must change the status of that Workspace to Submitted for Delivery to make it ready for the delivery process. After the status of the Workspace changes to Submitted for Delivery, it becomes a read-only Workspace and no more repository changes can be made to it until the merge process is complete.

To cancel the Workspace delivery process, thereby allowing repository changes to be made to Workspaces with a status of Submitted for Delivery, Workspace Administrators may revert the status of that Workspace back to Checkpointed and Editable by either:

- Using the Undo Submit for Delivery option in the Workspace menu, *or*
- Running the UndoSubmitForDelivery command from the Command Prompt window.

> **Note:** Only a Workspace Administrator who is also the owner of the MAIN Workspace can cancel the Workspace delivery process. If Workspace users or developers need to cancel a Workspace delivery process, they must submit the cancel request to the Workspace Administrator. After the request is approved, the Workspace Administrator will cancel the delivery process as requested.

To cancel the Workspace delivery process (Siebel Tools)

1. In the Workspace Dashboard, select Undo Submit for Delivery from the Workspace menu.
2. Click Yes on confirmation message that appears to revert the status of the Workspace from Submitted for Delivery to Checkpointed and Editable.

The following procedure shows how to cancel the Workspace delivery process using the siebdev command.

To cancel the Workspace delivery process using the siebdev command

1. Open a Command Prompt by clicking the Start button on your machine and then selecting the Run option.
2. In the Run window that opens, type in `CMD` in the Open field.

3. In the Command Prompt window that opens, change directory to the `<TOOLS_HOME>\BIN` folder as follows:

   ```
   cd <TOOLS_HOME>\BIN
   ```

   The Command Prompt returns a list of available arguments and parameters.
4. To cancel the Workspace delivery process, run the following command:

   ```
   siebdev /c tools.cfg /u SADMIN /p ******** /d ServerDataSrc /UndoSubmitForDelivery
   <Workspace Name>
   ```

   For example:

   ```
   $SIEBEL_HOME\BIN>siebdev /c tools.cfg /u SADMIN /p ******** /d ServerDataSrc
   /UndoSubmitForDelivery <Workspace Name>
   ```

5. After the status of the Workspace is reverted, end users and developers of that Workspace must close and then reopen the Workspace before they can make any repository changes on it.

Delivering Workspaces

You deliver modifications to implement your configuration changes in the user interface of the Siebel application. You deliver the modifications that you make to objects directly into the Siebel application.

Delivering a Workspace, or merging Workspace changes, applies the changes that are made in the Workspace into its parent Workspace. You must always perform the Workspace delivery process to the immediate parent Workspace. You cannot deliver the Workspace to any arbitrary Workspace.

If you are the owner of the parent Workspace or if you are a Workspace Administrator who is also the owner of the parent Workspace, then you can run the Workspace delivery process and deliver the Workspace in the following ways:

- **By clicking Deliver on the Workspace Toolbar.** Using this method, the Workspace Administrator and/or the Workspace owner can enter comments. For more information, see *Delivering Workspaces Using Workspace Toolbar*
- **By running the Deliver command from the Command Prompt.** Using this method, only Workspace Administrators can enter comments into the command line. For more information, see *Delivering Workspaces from the Command Line*.

  If you are a Workspace Administrator without being the owner of the parent Workspace, you can also use this method to deliver a Workspace.

Delivering a Workspace deploys the Workspace to the Runtime Repository tables.

Note: Workspaces must be delivered sequentially, rather than in parallel.

About Workspace Delivery Logging

Workspace delivery logging is located in two places:

- Merge information is delivered to a file named `WS_DELIVER_<WORKSPACE_NAME>.log`.

  This file contains Design Time Repository (DR) and repository merge information and, for Siebel Server setup, is located in the `<SIEBEL_ROOT>\log` folder.

- General information is logged in the `siebSrvr.log` file.

  This file contains Runtime Repository (RR) and publishing information and, for Siebel Server setup, is located in the `<SIEBEL_ROOT>\log` folder.

  For Siebel Tools, these logs are available in the `<SIEBEL_TOOLS>\log` folder.

Delivering Workspaces Using Workspace Toolbar

The following procedure shows how to deliver a Workspace using the Workspace Toolbar.

- When a Workspace is successfully delivered, the following options are disabled on the toolbar: Checkpoint, Revert, Submit, and Rebase.
- If Workspace delivery fails, then the following options are enabled on the toolbar: Rebase and Submit.

To deliver a Workspace using the Workspace Toolbar

1. In the *Workspace Dashboard*, click Deliver on the Workspace Toolbar.

   You typically select this option to deliver the changes in a Workspace to the MAIN Workspace.
2. In the Deliver Workspace dialog that opens, enter comments in the Enter Comment field and then click OK.

   Entering comments is mandatory when delivering a Workspace using the Deliver option from the Workspace menu/toolbar. The Deliver Workspace dialog also contains the following fields: From Workspace, To Workspace, Merge Status, and a Start Merge button.
3. Click Start Merge to trigger the merge process.

   An In Progress status bar appears and detailed information about the merge process appears in the Merge Status field.
4. Click Done when the merge process completes.

   The status and detailed information about the merge process appears in the *Workspace Dashboard*.

   If the merge process encounters a non-trivial problem or error, then the following message appears and the Workspace displays a status of Delivery Failed in the View Mode applet: `The delivery failed with error: Non-trivial merge found. Rbase needs to be done.`

   If you get this message, then you must rebase the Workspace, resolve the conflicts, and then try to deliver the Workspace again. For more information on how to rebase a Workspace, see *Rebasing Workspaces*.

Delivering Workspaces from the Command Line

The following procedure shows how to deliver a Workspace using siebdevcli from the command line. Note that siebdevcli is supported on both the server (Windows and UNIX platforms) and client.

To deliver a Workspace using the siebdevcli command

In the Workspace Explorer (in the *Workspace Dashboard*), confirm that the status of the Workspace you want to deliver is Submitted for Delivery.

1. Open a command line and then navigate to the `SIEBEL_HOME\BIN` folder.
2. Use the following command to run the siebdevcli.exe file:

```
siebdevcli /u user_name /p password /d odbc_datasource /l language_code
/c ReposiotryUpgrade.cfg /DeliverWorkspace "<Workspace Name>" WorkspaceDelivery >
```

For example:

```
:/siebel/mde/siebsrvr/bin/siebdevcli /u sadmin /p ******** /d "SIEBEL_DMGXYZ" /l ENU
/c "/persistent/ses/log/RepositoryUpgrade_20211001_204026/RepositoryUpgrade.cfg"
/DeliverWorkspace dev_sadmin_21_9_drop_15 WorkspaceDelivery >
/persistent/ses/log/RepositoryUpgrade_20211001_204026/Common/dummy.txt 2>
/persistent/ses/log/RepositoryUpgrade_20211001_204026/Common/dummy2.txt
```

The following procedure shows how to deliver a Workspace using the siebdev command.

To deliver a Workspace using the siebdev command

1. In the Workspace Explorer (in the *Workspace Dashboard*), confirm that the status of the Workspace is Submitted for Delivery.
2. Open a Command Prompt by clicking the Start button on your machine and then selecting the Run option.
3. In the Run window that opens, enter the value `CMD` in the Open field.
4. In the Command Prompt window that opens, change the directory to the `$SIEBEL_HOME\BIN` folder.

   The Command Prompt window returns a list of available arguments and parameters.
5. If you are the owner of the parent Workspace, run the following command to deliver the Workspace:

   ```
   siebdev /c tools.cfg /u <User ID> /p <password> /d ServerDataSrc
   /DeliverWorkspace "<Workspace Name>"
   ```

   For example:

   ```
   $SIEBEL_HOME\BIN siebdev /c tools.cfg /u SADMIN /p ******** /d ServerDataSRC_
   /DeliverWorkspace "<Workspace Name>"
   ```

   If you are a Workspace Administrator, you can optionally enter comments into the command line, as follows:

   ```
   $SIEBEL_HOME\BIN siebdev /c tools.cfg /u SADMIN /p ******** /d ServerDataSrc_
   /DeliverWorkspace "<Workspace Name>" "<Your Comment>"
   ```

   Note: The command line comments will appear in the comment fields in the Workspace Versions applet of the parent Workspace. If you do not provide comments in the command line, the system uses the description of the Workspace for the comment fields. If the description of the Workspace is not available, the system displays the value Enter Comments for these comment fields.

Workspace Delivery When Deploying Multiple Languages

For customers with multiple language deployments, all repository object types with a UI element (Applets, Views,or Screens) require a compiled Runtime Repository record for each deployed language. The compilation time required for this will be longer than for a single language instance. Workspace delivery, by default, will process each language in parallel to reduce the time required.

Recovering from a Failed Delivery

If any failure occurs during Workspace publish, then all committed transactions are rolled back automatically, including the Workspace merge. If rollback also fails, then the Workspace Administrator should manually initiate a Revert operation followed by a Deliver/Rebase operation again.

The WORKSPACE_STATUS (LOV type) includes the following LOV values to indicate *failure status*:

- **Publish Failed.** This status appears when rollback of publish fails during the Deliver operation.
- **Rebase Failed.** This status appears when rollback of publish fails during the Rebase operation.

Reasons for a Workspace publish failure during a Deliver/Rebase operation include the application crashes, the database goes down or the database connection is lost. For example, if the database connection is lost, then the Deliver/Rebase operation may terminate abruptly or erroneously leaving the Integration Workspace in an incomplete state. In such a case, the Integration Workspace may end up in an Edit in Progress, Publish Failed, or Rebase Failed status. The failed Deliver operation would have generated a new version on the Integration Workspace, which is a bad version and needs a manual rollback. In this incomplete state, no other operation is allowed on the Integration Workspace except Revert. A Workspace Administrator must manually initiate a Workspace Revert operation on the Integration Workspace to roll back the incomplete Deliver/Rebase operation. This will revert the Integration Workspace back to a proper state. Thereafter, you may try to perform the Deliver/Rebase operation again.

Comparing Workspace Versions

Note: This tasks applies only to Siebel Tools.

You can use the Compare option in the Workspace menu to identify the changes that are made to the objects in two selected Workspace versions.

To compare Workspace versions

1. In the Workspace Dashboard, select the Compare option from the Workspace menu.

   The Compare Workspace dialog that opens contains the following group boxes: First Selection and Second Selection. In the First Selection group box, the Workspace Name and Version fields are populated with the current (open) Workspace name and version .
2. In the Second Selection group box, specify the second Workspace name and version:

   a. Select a Workspace name from the Workspace Name drop-down list.
   b. Select a version from the Version drop-down list (which includes all versions of the selected Workspace.)
3. Click OK.

   The status bar indicates that the process is running. When the process finishes, the Compare Objects dialog opens, displaying all top-level objects that have differences in the two selected Workspace versions.
4. Select one object that has such differences.

   The Compare Objects dialog shows the object properties, including the similarities and the differences for each attribute.
5. Click Close to close the Compare Objects dialog.

Adding the Workspace Prefix System Preference

To ensure all users follow standards while creating Workspace names, a Workspace Administrator who is also the owner of the MAIN Workspace needs to create a new system preference called Workspace Prefix.

When users create Workspaces that do not have values for the Workspace Prefix system preference, the default prefix values are set as in the application.

To add the Workspace Prefix system preference

1. Go to the System Preferences view and query for the `Workflow Prefix` system preference.

   If this system preference is not found, then create a new record named `Workflow Prefix`.
2. In the System Preference Value column, specify the desired prefix characters.

   > **Note:** Do not include spaces or non-alphanumeric values in the prefix.

For more information about setting system preferences, see *Siebel Applications Administration Guide* .

Configuring Non-Workspace Objects

The following object types and their child objects are non-Workspace objects:

- Dock Objects
- EIM Interface Table
- Project
- Schema maintenance
- Server components
- Repository
- Table
- Type
- Workflow Policy Column
- Workflow Policy Object
- Workflow Policy Program

Only one single public version of the object is available for all users. Hence, configuration on instances of this object is centrally controlled by the Workspace Administrator, who can lock all relevant projects and objects. Developers are able to configure the respective objects or projects after their requests to release the locks on these objects or projects are granted by the Workspace Administrator.

When developers configure non-Workspace objects, they must develop and test the non-Workspace objects against a dedicated environment and deliver the changes (or import using the SIF-OUT/SIF-IN command) directly to the master database by requesting that the Workspace administration team unlock the project.

For changing the existing database schema or adding new tables to the database schema, all users must follow the same process that is currently followed by all development teams; that means, respective teams need to cooperate with the Workspace Administrators for any change that is required in the database schema of their applications.

For Workflow and Task configurations, developers must ensure that only the `0th` version on the standard database exists.

> **Note:** The project must be locked for non-Workspace objects before the object can be modified. The locking and unlocking of projects can be controlled by an administrator.

Workspace Enabled Admin Data

The Siebel application enables certain data types (called Admin Data) to participate in Workspace life cycles just as Repository objects such as Applets and Business Components do.

You create/update/delete the records in the end user application and deliver, rebase, migrate the data exactly as you would with Web Tools.

The current Admin Data types that are available to participate in Workspaces are:

- LOVs (enabled by default)
- Predefined Queries
- Data Maps
- EAI Data Maps
- Responsibilities (and their relationship to Views)

Enabling Admin Data Types for Workspaces

You must enable each type of Admin Data that you wish to use in Workspaces. LOVs are automatically enabled, but the other types of Admin Data require a that the schema and underlying database be updated to support Workspaces. All types except LOVs are optional. Enabling Admin Data types is done in Web Tools by a user with the Workspace Administrator Responsibility.

To Enable Admin Data Types

To Enable Admin Data Types in Web Tools follow the steps given below:

1. Go to **Tools**→ **Workspace Enable Admin Data**.
2. In the **Admin Data Types Applet**, you can select the type of Admin Data you wish to enable. You can select more than one type at a time.
3. When you select an Admin Data Type, the tables that will be affected show in the **Affected Tables Applet**. This informs you of the Tables that will change.
4. The Workspace Enabled check box shows if a particular Admin Data Type has already been enabled. **List Of Value** will automatically be checked because PostInstallDBSetup will already have enabled it. The check box is read only.
5. Once you have chosen all the Admin Data Types you want to enable you have two options.

   Note: These actions cannot be undone without restoring the database.

   a. **Apply Schema** – This button applies the schema and physical database changes to the affected tables. When you click this button, the database parameters dialog will display. Fill in the data and click OK to start the process. After a few minutes the progress bar will disappear, and the data type will be enabled.
   b. **Generate Schema**– This button applies the schema changes to the affected tables so that the correct SQL can be generated for application to the physical database. Use this option if someone other than the one using Web Tools to generate the Schema changes will apply them to the database. For example, sometimes only the Database Administrator (DBA) is allowed to update the physical database. In that case, one person generates the Schema changes and the DDL files by clicking the Generate Schema button and the DBA applies the SQL files to the physical database.

c. **Update Repository–** Use this button only if you used the Generate Schema button and a DBA has applied the changes to the physical database. Once those changes are complete, come back to Web Tools to finish the process by clicking this button.

Once these steps are complete, the Admin Data Types that you have enabled are ready to use in the end user application such as Call center.

Note: Once you enable an Admin Data type to participate in Workspaces, you cannot undo this action without restoring your development database to its state before you enabled the Admin Data type.

Note: In your design Repository, while in the process of Workspace enabling any Admin Data type (the process is not complete) avoid changing any of the Admin Data types. For example, you have chosen the EAI Data Map Admin Data Type and have clicked the Generate Schema button. The database administrator has not yet applied the SQL to the physical database so the Workspace Enablement of the EAI Data Map type is incomplete. During this time, do not change records in the EAI Data Map Admin Data type.

Similarly, while the IRR migration that will enable Workspaces on your Target Repository is in progress, do not make changes to the Admin Data types that will be enabled once the IRR completes.

Using Workspaces for Seed Data

Workspaces are also used for certain types of seed data. This feature enables developers to have their work neatly isolated. It also maintains the changes that are set at the design time and the runtime as a single package. This feature requires no downtime of the seed data to deploy.

Using Workspaces for the seed data, note that:

- By default, Oracle Siebel delivers seed data Workspace-enabled for LOVs and you cannot disable those Workspaces on seed data.
- The concept of Workspace for seed is similar to the concept of Workspace for metadata. The only difference is metadata changes are specific to the user whereas seed data changes apply to all users under the nearest integration branch.
- Modifying the seed data occurs in the closest integration branch; hence, all users under the same integration branch can see each other’s changes immediately.
- Seed data is copied automatically when a user creates a new integration branch, so there is a separate copy of the seed data for every integration branch, including the MAIN branch.
- Modifying seed data does not create the Workspace version. You must always modify metadata to create the Workspace versions.
- Workspace delivery from the user-to-integration branch carries only metadata modifications because seed data modification exists in the integration branch.
  However, Workspace delivery from Integration-to-Integration Workspace carries both seed data and metadata changes.
- You can access Siebel Tools or the thick client using the `\editseeddata` option to create a corporate record and then start using the Workspace.
  For more information on how to use the `\editseeddata` option, see Siebel Developer's Reference Guide.
- You cannot perform the Revert process for the changes in the seed data.

For more information on how to use database utilities to migrate metadata and seed data, see Siebel Database Upgrade Guide.

To change seed data

1. Create a user Workspace under the integration branch.
   For more information on how to create new Workspaces, see *Creating New Workspaces*.
2. Modify the seed data and metadata as needed.
   The modification of seed data in the user Workspace causes the modification in the nearest integration branch.
3. Perform the Checkpoint process on the Workspace.
   For more information on how to perform the Checkpoint process, see *Performing the Checkpoint Version Process*.
4. Submit the Workspace for delivery.
   For more information on how to submit the Workspaces for delivery, see *Submitting Workspaces for Delivery*.
5. Deliver the Workspace to the nearest integration branch.
   For more information on how to deliver Workspaces, see *Delivering Workspaces*.
6. Deliver the Integration Workspace to the MAIN branch.
   For more information on how to deliver the Integration Workspace to the MAIN branch, see *Delivering Workspaces*.
7. If there is non-trivial change or conflict in the seed data, perform the following steps to rebase and deliver the Workspace:
   a. Rebase the Workspace.
      For more information on how to perform the Rebase process, see *Rebasing Workspaces*.
   b. Select the original value to override the current Workspace value; otherwise, use the default new value.
      For more information on how to fix the conflict errors and apply resolutions, see *Detecting Conflicts in Workspaces and Applying Resolutions*.
   c. After the Rebase process is completed, deliver the Workspace again.
      For more information on how to deliver the Workspace, see *Delivering Workspaces*.

Example: Inserting Seed Data Records

1. Create the Integration Workspace called INT_CRM2017 under the MAIN Workspace.
   For more information on how to create new Workspaces, see *Creating New Workspaces*.
2. Create another Workspace called EXAMPLE_WS under the INT_CRM2017 Workspace.
3. Optionally, modify the metadata record.
4. From the menu bar in the Siebel Tools window, choose the Screens menu, System Administration, and then select the List of Values menu item.
5. Change the Type value to CUT_ACCOUNT_TYPE.
6. Change the display name from BILLING to BILLINGNEW.
   The new display name BILLINGNEW is available in the INT_CRM2017 Workspace.
7. Perform the Deliver process to deliver the EXAMPLE_WS Workspace to the INT_CRM2017 Workspace.
   For more information on how to deliver Workspaces, see *Delivering Workspaces*.

8. Perform the Deliver process again to deliver the INT_CRM2017 Workspace to the MAIN Workspace.
9. Open the MAIN Workspace.
10. Navigate to the Account List view and select the Account Class drop-down list.

    The BILLINGNEW value is listed in the Account Class drop-down list. If the BILLINGNEW value is not listed, you must clear the cache by navigating to `System Administration\Data\List of Values` and click the Clearcache button.
11. Check to ensure that both INT_CRM2017 and MAIN Workspaces now list the BILLINGNEW value.

Publishing Tables

Tables can also be published into the S_RR_TABLE table using the same concept by which other metadata objects are published into the Runtime Repository table. A table maintains a different version number and it is tracked from the S_RR_OBJ_ITEM.TBL_VER column.

However, table versioning is different from other metadata versioning because a table is not Workspace-enabled. While performing the Full Publish process, the table version is always 0 (zero), irrespective of the Workspace version.

A table can be published in two ways using Siebel Tools in a Windows environment:

- Using the Apply/DDL button.
- Using the siebdev.exe utility.

Publishing Table in Source-to-Target Migration

In the source environment, a table needs to be published either by using the Apply/DDL button or by running the siebdev.exe utility command.

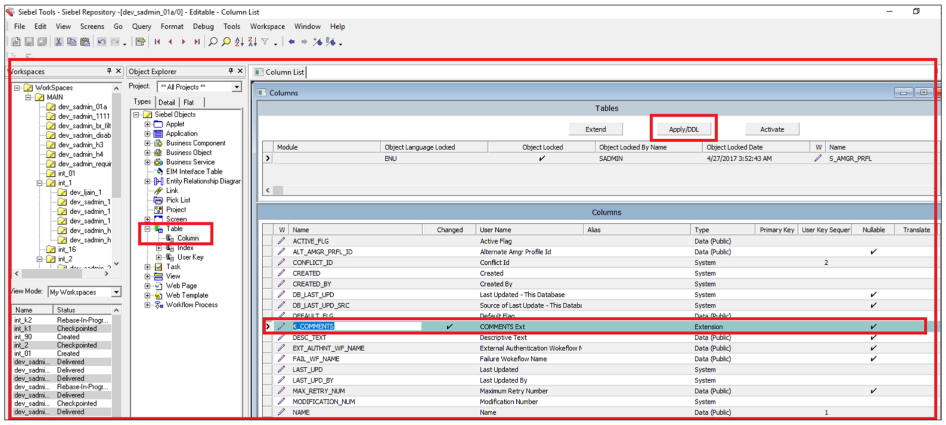
In the target environment, publishing a table is generated by the Incremental/Full source-to-target Runtime Repository migration. Hence, a table cannot be published explicitly in the target environment.

To publish a table using the Apply/DDL button (Siebel Tools)

1. Update the required table schema changes as needed.

2. Search for the table that you want to publish.

   The following image displays the S_AMGR_PRFL table and its corresponding Columns. The Apply/DDL button is available on the applet banner.



Name	Status
int_k2	Rebase-In-Progr...
int_k1	Checkpointed
int_90	Created
int_2	Checkpointed
int_01	Created
dev_sadmi...	Delivered
dev_sadmi...	Delivered
dev_sadmi...	Delivered
dev_sadmi...	Rebase-In-Progr...
dev_sadmi...	Delivered
dev_sadmi...	Checkpointed
dev_sadmi...	Delivered

Module	Object Language Locked	Object Locked	Object Locked By Name	Object Locked Date	W	Name
	ENU	✓	SADMIN	4/27/2017 3:52:43 AM		S_AMGR_PRFL

W	Name	Changed	User Name	Alias	Type	Primary Key	User Key Sequer	Nullable	Translate
	ACTIVE_FLG		Active Flag		Data (Public)				
	ALT_AMGR_PRFL_ID		Alternate Amgr Profile Id		Data (Public)			✓	
	CONFLICT_ID		Conflict Id		System		2		
	CREATED		Created		System				
	CREATED_BY		Created By		System				
	DB_LAST_UPD		Last Updated - This Database		System			✓	
	DB_LAST_UPD_SRC		Source of Last Update - This Databa		System			✓	
	DEFAULT_FLG		Default Flag		Data (Public)				
	X_COMMENTS	✓	COMMENTS Ext		Extension			✓	
	DESC_TEXT		Descriptive Text		Data (Public)			✓	
	EXT_AUTHNT_WF_NAME		External Authentication Wokeflow N		Data (Public)			✓	
	FAIL_WF_NAME		Failure Wokeflow Name		Data (Public)			✓	
	LAST_UPD		Last Updated		System				
	LAST_UPD_BY		Last Updated By		System				
	MAX_RETRY_NUM		Maximum Retry Number		Data (Public)			✓	
	MODIFICATION_NUM		Modification Number		System				
	NAME		Name		Data (Public)		1		

3. Click the Apply/DDL button.

   The Choose Option dialog opens where you specify either to apply the schema changes to the database or create the DDL file.
4. Select either Apply or Generate DDL, and then click OK.

   - If you select Apply, then the Apply Schema window opens where you can specify the table, database, and DDL details. When you click OK, the schema is applied to the tables and the tables are republished with the next version number. The table compiled-object definitions are present in the S_RR_TABLE table in the next table version, and the list of effected tables is present in the S_RR_OBJ_ITEM table.
   - If you select Generate DDL and then click OK, the table will not be published; however, it will generate the DDL in the specified location.

To publish a table using the siebdev.exe utility

1. Open a Command Prompt window on your machine by clicking the Start button and selecting the Run option.
2. In the Run window, enter the `CMD` value in the Open field.
3. At the Command Prompt, run the following *siebdev.exe* utility command:

```
siebdev /c tools.cfg /l <LANG> /d <ServerDataSrc> /u <username> /p <password> /
IncrementalTablePublish <FileName>
```

   The contents of `<FileName>` can be `<TableName>:<Operation>` where the operation can be Insert, Update, or Delete. These are the examples of the `ddl_TableSchema.txt`:

   - `S_RR_OBJ_VER:Update`
   - `S_RR_TEST1:Insert`
   - `S_RR_TEST2:Delete`

> **Note:** The `ddl_TableSchema.txt` file is always auto-generated when the schema is applied using DDLIMP. This file is generated in the same location as the log file under the file name `<log_File_Name>_TableSchema.txt`. The name of the generated file cannot be changed.

Workspaces Administration

This topic describes how to perform some Workspace administrative tasks. These tasks are performed only by the users who are assigned the user role Workspace Administrator. This topic includes the following information:

- *Flattening Workspace Versions*
- *Executing the Full Publish Process*
- *Enabling Workspaces*

Flattening Workspace Versions

A Workspace Administrator who is also the owner of the MAIN Workspace (root, parent, or master Workspace) can perform the process of flattening Workspace versions using the Command Prompt. No other users can perform this process.

The flattening of Workspaces collapses the MAIN branch to a single version with all the latest changes and subsequently deletes all the child Workspaces.

The only time that flattening Workspaces is required is after the addition of a new language using the Siebel Database Configuration Wizard and before MLOV conversion. There are no other situations that require a flattening Workspace operation.

When flattening a Workspace before an MLOV conversion, the sequence of events is as follows:

1. Add a new language to the database.
2. Deliver Workspaces that have been completed and tested and MAIN.
3. Archive any Workspaces which contain in progress development work so that they can be re-imported into a Workspace after the FullPublish completes.
4. Run FlattenWorkspace, see the following procedure.
5. Run MLOV conversion (which you must do for the new language to work correctly).
6. Run EnableWorkspace, see *Enabling Workspaces*.

   > **Note:** The EnableWorkspace utility should only be run in the case where a new language has been added as part of this process. There is no other time that the EnableWorkspace utility should be run.

7. Run FullPublish, see *Executing the Full Publish Process*.

Note: Although there is no requirement to do so, it is possible to flatten Workspaces at any time, for example, to clear out all old Workspaces. However, this is not typically recommended for the following reasons:

1. It will collapse all Workspaces, even those that have not been delivered. If there are other integration branches or Developer Workspaces in which active parallel development is taking place, those would be lost. The recommended approach to clean up old integration branches is to delete them see *Deleting Integration Workspaces*.
2. After a flatten Workspace operation, it is required to run Full Publish for these two scenarios. You do not have to run Full Publish if you are flattening Workspaces just to clean up.

   a. You have added a new language to the application as specified above.
   b. You are performing an IRM upgrade.

   After running Flatten Workspace, which in turn requires that the next migration to any downstream Runtime Repository Database be a Full Migration. That is, it eliminates the possibility for incremental migration.

CAUTION: In order for the flattening process to run successfully, the parameter ServerDbODBCDataSource under the `[Siebel]` section in the Tools.cfg file must point to the correct data source.

The following example shows the setting of the parameter ServerDbODBCDataSource under the `[Siebel]` section in the tools.cfg file:

```
[Siebel]
…
ReportsODBCDataSource = Siebel Reports: Access
LocalDbODBCDataSource = m:/siebel Local DB
ServerDbODBCDataSource = "ORAJQ141"
DockRepositoryName = Siebel Repository
HoldExportOdbcConnection = FALSE
…
```

To flatten the Workspace versions using the Command Prompt

1. Open the Command Prompt window on your machine by clicking the Start button and selecting the Run option.
2. In the Run window that opens, enter the value `CMD` in the Open field.
3. In the Command Prompt window that opens, change directory to the `$SIEBEL_HOME\BIN` folder.
4. Run the following flatten Workspace command:

   ```
   siebdev /c tools.cfg /u <User ID> /p <password> /d ServerDataSrc /FlattenWorkspace
   ```

   Or, to flatten a specific Integration Workspace hierarchy:

   ```
   siebdev.exe /c tools.cfg /l enu /d ServerDataSrc /u <User ID> /p <password> /flattenWorkspace
   <INTEG NAME>
   ```

   For example:

```
$SIEBEL_HOME\BIN>siebdev /c tools.cfg /d ServerDataSrc /u sadmin /p ********
/FlattenWorkspace
```

or

```
 $SIEBEL_HOME\BIN>siebdev.exe /c tools.cfg /l enu /d ServerDataSrc /u sadmin /p ********
/flattenWorkspace "Int_Oct_2024"
```

5. Confirm that the flatten Workspace version process shows the process is successfully completed.

After flattening Workspaces, you must run the Full Publish process if you have added a new language or are performing an IRM upgrade. You do not have to perform a Full Publish if you have simply flattened a Workspace of other reasons. For more information, see *Executing the Full Publish Process*.

Executing the Full Publish Process

Before running the Full Publish process, as described in the following procedure, note the following about the Full Publish process:

- The Full Publish process must be performed only in Siebel Tools; it cannot be performed in Web Tools.
- Not all repository objects are runtime-enabled; only the repository objects that can be compiled are runtime-enabled.
- The only time that you need to manually run Full Publish is after a flattening Workspace operation - running Full Publish is not required at any other time. For more information on how to flatten a Workspace, see *Flattening Workspace Versions*.
- After running Full Publish, it will not be possible to run incremental migration to any downstream Runtime Repository environment until a Full Migration has been executed to that environment.

To run the Full Publish process

1. Open the Command Prompt window on your machine by clicking Start and selecting the Run option.
2. In the Run window, enter the value `CMD` in the Open field.
3. At the Command Prompt, run the Full Publish process as follows:

```
siebdev /c tools.cfg /TL <lang_code> /d <dataSource_name> /u <username>
/p <password> /FullPublish
```

   - The following example command runs `FullPublish` in the database with one language, English (ENU):

   ```
   siebdev /c tools.cfg /TL ENU /d ServerDataSrc /u sadmin /p ******** /FullPublish
   ```

   Notice that the `/d` parameter in this example refers to the datasource defined in the tools.cfg file.

   Note also that the `<lang_code>` parameter can either be a single language or multiple languages. You must specify the language that needs to be published in the `/TL` parameter. For more information about

Runtime Repository business components that are language independent, review the information in the following table.

- The following example command runs `FullPublish` in the database with one language, German (DEU):

  ```
  siebdev /c tools.cfg /TL DEU /d ServerDataSrc /u sadmin /p ******** /FullPublish
  ```

- The following example command runs `FullPublish` in the database with three languages: English (ENU), German (DEU), and Japanese (JPN):

  ```
  siebdev /c tools.cfg /TL ENU,DEU,JPN /d ServerDataSrc /u SADMIN /p ******** /FullPublish
  ```

After the Full Publish process completes successfully, you can launch the Siebel application only in those languages that have the application strings and languages that are specified in the Full Publish process. In case you need to launch the Siebel application in any additional language, run Full Publish again by specifying those languages.

In the last example, Siebel application is already published in ENU, DEU, and JPN. However, if their is a requirement for FRA (French), then you need to run Full Publish again specifying all the required languages:

```
siebdev.exe /c tools.cfg /TL ENU,DEU,JPN,FRA /d ServerDataSrc /u sadmin /p ******** /FullPublish
```

If you need to drop a language, then repeat the Full Publish process. For example, you can remove JPN as follows:

```
siebdev.exe /c tools.cfg /TL ENU,DEU,FRA /d ServerDataSrc /u sadmin /p ******** /FullPublish
```

Full Publish will recompile all the objects again and you can start Siebel later in the three remaining languages.

> **Note:** You cannot add or drop a new language through Incremental Publish; you can do so only through Full Publish. During Full Publish, Siebel will be down.

About Using FullPublishInternal

When you run the `FullPublish` process for multi-language deployments, it will open a session for each language and run parallel processes for each language. This can result in performance, memory, and CPU issues. As an alternative, the Workspace Administrator can use the `FullPublishInternal` flag to force sequential (rather than parallel) processing.

This is done by first running `FullPublish` as normal with your primary and any secondary language, then running `FullPublishInternal` for each additional language.

For example, if you have 10 languages deployed:

1. Run `FullPublish` as normal for two of your required languages as follows:

   ```
   siebdev /c tools.cfg /TL ENU,DEU /d ServerDataSrc /u <username> /p <password> /FullPublish
   ```

   This takes care of the groundwork that is needed (for example, ensures that you have already run FlattenWorkspace) and creates some appropriate system records that are needed behind the scenes.
2. Then, for each additional language needed, run `FullPublishInternal` sequentially as follows:

   ```
   siebdev /c tools.cfg /TL FRA /d ServerDataSrc /u <username> /p <password> /FullPublishInternal
   siebdev /c tools.cfg /TL JPN /d ServerDataSrc /u <username> /p <password> /FullPublishInternal
   siebdev /c tools.cfg /TL HEB /d ServerDataSrc /u <username> /p <password> /FullPublishInternal
   . . .
   . . .
   ```

Enabling Workspaces

Note: This task applies to Siebel Tools and Web Tools.

Workspaces are enabled automatically during installation and upgrade.

Workspace Administrators can enable Workspaces by running the executable (EnableWorkspace) utility. After Workspaces have been enabled, the root Workspace is added to the S_WORKSPACE table and a repository-based version record is added to the S_WS_VERSION table.

For more information on the responsibilities of a Workspace Administrator, see *About Workspaces*.

Note: The EnableWorkspace utility and its command-line arguments are applicable for all platforms. You can call EnableWorkspace.exe or EnableWorkspace (without .exe) for the Windows platform; however, you must call EnableWorkspace (without .exe) for other non-Windows platforms.

To enable Workspaces

1. Execute the EnableWorkspace utility without any arguments to get a list of the supported options:

   - `-s`: Siebsrvr/Tools Installation path specified (required)
   - `-t`: Siebel Table Owner (required)
   - `-u`: TBLO Username (required)
   - `-p`: TBLO Password (required)
   - `-o`: ODBC Data Source (required)
   - `-d`: DB Platform Name (Oracle, MSSQL, DB2UDB or DB2390)
   - `-l`: Log File Name (default: EnableWorkspace.log)
   - `-w`: Workspace Owner Username (required)
   - `-r`: Repository Name (default: "Siebel Repository")
   - `-a`: Action ((R)epository, (S)eed, (B)oth) (default: (B))
   - `-b`: RepositoryType ((R)unTime, (D)esignTime, (B)oth) (default: (B))
   - `-v`: Generate Schema File Only (Default : N, for DB2390 only)
   - `-f`: Seed inp file
   - `-y`: Siebel UserName (required)
   - `-z`: Siebel User Password (required)
   - `-i`: Integration Branch (default: All integration branches unless passed integration branch name with this parameter)
   - `-c`: Workspace Checkpoint Status (default: N)
   - `-k`: Log directory (default: "Current Directory ")
   - `-e`: Language Code

2. Run the EnableWorkspace utility using the available arguments and parameters, for example, as follows:

```
<-- ORACLE DB -->
C:\Siebel\siebsrvr\BIN> EnableWorkspace -s "C:\Siebel\siebsrvr" -u SIEBEL -p ********
-t SIEBEL -r "Siebel Repository" -d "Oracle" -o ORAXXXX -w SADMIN -y SADMIN -z ********
```

   Or:

```
<-- MSSQL DB -->
C:\Siebel\siebsrvr\BIN> EnableWorkspace -s "C:\Siebel\siebsrvr" -u SIEBEL -p ********
-t dbo -r "Siebel Repository" -d "MSSQL" -o Siebel_DSN -w SADMIN -y SADMIN -z ********
```

3. Review the output stages, representing the progress of the tasks performed by the utility.

   The Command Prompt window shows the following output stages after you run the EnableWorkspace utility:

   - Stage 1 of 8: Setting the Repository ID
   - Stage 2 of 8: Validate the Workspace Data
   - Stage 3 of 8: Generate the Workspace Data
   - Stage 4 of 8: Update the Workspace Data in all Repository Tables
   - Stage 5 of 8: Update the indexes
   - Stage 6 of 8: Update the Workspace Data in all Seed Tables
   - Stage 7 of 8: Update the statistics for all Workspace enabled Repository/Seed Tables
   - Stage 8 of 8: Post Run Cleanup

4. Confirm that the EnableWorkspace utility has processed successfully on your environment by doing the following:

   a. Access and open your client.
   b. In the Workspace Explorer, confirm that there is one and only one Workspace named MAIN.

6 Parallel Development Using Workspaces

Parallel Development Using Workspaces

This chapter describes the Parallel Development Using Workspaces feature, including how to administer and use the feature. It includes the following topics:

- *Administering Parallel Development Using Workspaces*.
- *Using Parallel Development Tools for Workspaces*

Note that the tasks involved in this feature are performed only by users who have the permissions to use the Parallel Development Using Workspaces feature.

Administering Parallel Development using Workspaces

Parallel development enables Siebel Workspace users to create and to work on integration branches. This is useful when users work in a parallel development environment. Parallel development maintains the relationship between the design time (Workspaces) and the runtime repository tables even after the parallel development feature for Workspaces is enabled. Using the parallel development feature for Workspaces, you can create or open Workspaces from any parent or version Workspace and get a consistent manifestation of the runtime metadata.

This section describes how to administer the *parallel development using Workspaces* feature. Only users with the correct administrative permissions can perform these tasks, which include the following:

- *Enabling the Parallel Development Feature for Workspaces*.
- *Understanding Integration Workspaces*.
- *Creating Integration Workspaces*.
- *Setting an Integration Workspace as the Default Workspace Branch*.
- *Setting System Preferences for Parallel-Development Workspaces*.
- *Launching the Web Client with a Specified Workspace*.
- *Deleting Integration Workspaces*
- *Difference Between Deleting Development and Integration Workspaces*

For more information on how to detect errors and conflicts in parallel-development Workspaces, and then apply a resolution, see *Detecting Conflicts in Workspaces and Applying Resolutions*.

Enabling the Parallel Development Feature for Workspaces

Only users that have administrative permissions can enable the parallel development feature for Workspaces. In general, Siebel administrators perform this task for all members in a development team.

Workspaces must be enabled before you enable the parallel development feature for Workspaces in your working environment. For more information on how to enable Workspaces, see *Workspaces Administration*. For more information on how to use the Workspaces feature as a Siebel developer or user, see *Using the Command Line*.

To enable the Parallel Development feature for Workspaces

1. As an administrator and owner of the development environment, log in to the Siebel application.

   > **Note:** You can set the user preferences using either the Siebel application or Siebel Web Tools. For changes to take effect in Siebel Web Tools, you must log out of the application and then log back in again. For changes to take effect in the Siebel application, you must restart the services of the application.

2. Select System Preferences from the Tools menu to open the System Preferences view.
3. Set the following System Preference parameter to Y (yes):

   ```
   Enable Integration Workspaces=Y
   ```

4. After setting the parameter Enable Integration Workspaces to Y, restart the services of Siebel Server Application to reflect the new parameter value on the server side.

   The environment is now ready for all users to create the parallel-development Workspaces.

   For information about restarting Siebel Servers, see *Siebel System Administration Guide* .

Understanding Integration Workspaces

Users with the administrative permissions can create parallel-development Integration Workspaces. Developers use these Integration Workspaces to set up a parallel release development environment. The names of all Integration Workspaces start with the prefix `int_` and these Workspaces have the corresponding run-time repository version.

Only administrators can perform the following Workspace operations on Integration Workspaces: Submit for delivery, Rebase, Deliver, and Undo Submit for delivery. Also, only administrators can deliver the changes in Integration Workspaces to the immediate parent Integration Workspace. The rebase process must be performed from the immediate parent Integration Workspace.

For more information on how to deliver Workspace changes and rebase Workspaces, see *Using the Command Line*.

> **Note:** Users with the developer permissions can create Development Workspaces but they cannot create Integration Workspaces nor deliver their Workspace changes to the multiple-release Integration Workspaces simultaneously. In fact, they can open the Integration Workspaces and view the version changes in read-only mode, but they cannot perform any other Workspace-related actions. After the administrators successfully deliver their Workspace changes into the Integration Workspace, developers can open the Integration Workspaces in the Siebel application and view the latest delivered changes.

For more information on how to create Development Workspaces, see *Understanding Development Workspaces* and *Creating Development Workspaces*.

After the Merge process from the source Workspace to the destination Workspace has completed, the Publish process runs automatically if the destination Workspace is an Integration Workspace. The publish process is not initiated if the destination Workspace is a Development Workspace. When the Publish process starts, processing updates all Runtime Repository tables with the latest metadata values. After the Publish process has successfully completed, users can see the latest compiled and published changes when they open the Integration Workspace in the Siebel application.

To avoid conflicts, it is recommended that you always create Workspaces branching from the latest versions of the Integration Workspaces. For more information on how to resolve the conflicts in the emerging resolution window, see *Detecting Conflicts in Workspaces and Applying Resolutions*.

The parallel-development Workspaces are incremented either when:

- The child-level Integration or Development Workspaces merge their object repository changes with them, or
- Users perform the rebase operation on an Integration Workspace to uptake the changes that are made in the parent Integration Workspace (which means the Check Point option is not available for all users, including the administrators, to manually checkpoint the version of the Integration Workspaces).

Note: You cannot perform the Revert functionality on the Integration Workspace.

Creating Integration Workspaces

The following procedure describes how administrators create an Integration Workspace.

Note: Only administrators can create Integration Workspaces.

To create an Integration Workspace

1. In the Workspace Explorer (in the *Workspace Dashboard*), select the parent Integration Workspace that you want.
2. Click Create on the Workspace Toolbar:

   The Create Workspace window that opens contains the following elements:

   a. Name of the selected parent Workspace.
   b. Integration Workspace option. This option is available only for administrators (it is unavailable for other users, including developers.
   c. Comment field.
   d. Cancel and Create Workspace buttons.
3. Select the Integration Workspaces option.

   In the Name field, the prefix of the Workspace name changes to `int_`.
4. Append the Workspace name text to the prefix `int_`.

   For example: `int_cir4build`.
5. (Optional) Enter a comment in the Comment field.
6. Click Create Workspace to complete the Workspace creation.

Setting an Integration Workspace as the Default Workspace Branch

When users access the Siebel application after they set an integration branch as the default Workspace branch, the system loads the application based on the latest metadata changes of the Integration Workspaces that they have set as the default Workspace instead of the values in the MAIN Workspace. This setting enables the Siebel application to display the latest metadata values that are stored in the run-time repository tables. Every time the Integration Workspace is incremented by one version, the latest metadata values are stored in the run-time repository tables.

When the application is reloaded, changes are reflected in the Siebel application window based on the latest version of Integration Workspace.

Administrators can set any Integration Workspace as their default Workspace branch where their changes do not impact other users who are branched out from the other Integration Workspaces. Meanwhile, developers can work in parallel on multiple features of Siebel application without disturbing other features. Moreover, developers can verify how their latest metadata configuration appears in the Siebel application window without delivering their changes to the MAIN Workspace.

To set up an Integration Workspace in the environment as the default Workspace branch, administrators must update the parameter WorkspaceBranchName at the component level as shown in the following example. Notice that administrators must run this parameter by connecting to the server manager:

Command:

```
change param WorkspaceBranchName=<Workspace_name> for comp <OBjMgr_lang>
```

Example:

```
change param WorkspaceBranchName=int_ip2021rel for comp SCCOBjMgr_enu
```

To reflect the changes, administrators must restart the services of the Siebel server.

Setting System Preferences for Parallel-Development Workspaces

The following table describes the system preferences that must be set if using parallel development for Workspaces.

System Preference	Description
Enable Integration Workspaces	Administrators set this preference to Y to enable parallel development for Workspaces. The default value for this preference is N. After a parallel development for Workspaces merge is successful, the default value for this preference will be Y. When an administrator changes the Enable Integration Workspace value, services must be restarted for the changes (the latest Integration Workspace preferences on the server side) to take effect.
Enable Tools Workspace Deliver	Administrators set this preference to Y (default value) to deliver and publish Workspace changes. If you do not want to deliver Workspaces, then set this preference to N. When an administrator changes the Enable Tools Workspace Deliver value, services must be restarted for the changes (the latest Integration Workspace preferences on the server side) to take effect.
Integration Workspace Prefix	Integration Workspaces start with the prefix value `int`, which is derived from the Integration Workspace Prefix preference. If required, the administrator can set up a new Integration Workspace prefix value. When an administrator changes the Integration Workspace prefix value, services must be restarted for the changes (the latest Integration Workspace preferences on the server side) to take effect.
Workspace Admin Responsibility	This responsibility is granted only to Workspace administrative users who will have the full privileges to manage the MAIN/Integration Workspace.

System Preference	Description
	For more information about the detailed tasks of Workspace Administrators, see *Workspaces Administration*.
Workspace Prefix	Developer Workspaces start with the prefix value **`dev`**, which is derived from the Workspace Prefix preference. If required, administrators can set up a new Workspace prefix value – see *Adding the Workspace Prefix System Preference*. When an administrator changes the Workspace prefix value, services must be restarted for the changes (the latest Workspace preferences on the server side) to take effect.

Launching the Web Client with a Specified Workspace

To launch the web client with a specified Workspace, the administrator must update the `[Workspace]` and `[ingraObjMrg]` sections in the configuration file of the Siebel application.

For example, in the Siebel Financial application, they must update the `[Workspace]` sections in the fins.cfg file, which is the same for other applications. Administrators must update these sections in the respective applications' configuration files.

```
[Workspace]
Name = int_ip2017
Version = Latest

[InfraObjMgr]
RepositoryType = RUNTIME
```

Deleting Integration Workspaces

Users with the Workspace Administrator responsibility can delete Integration Workspaces. When you delete an Integration Workspace, the corresponding design time data, run time data, and seed records are deleted, including any associated development and Integration Workspaces. Also, any reference to the Integration Workspace in merge or rebase reports in other views will show as *deleted*.

Note: The MAIN Workspace cannot be deleted.

The following procedure shows how to delete an Integration Workspace. For more information, see *Difference Between Deleting Development and Integration Workspaces* and *Deleting Developer Workspaces*.

To delete an Integration Workspace

1. In the Workspace Explorer (in the *Workspace Dashboard*), select the Integration Workspace that you want to delete.

   > **Note:** Select a Workspace that is not active or is not configured for migration because you cannot delete an active Workspace or perform migration once the source Workspace is deleted. Ensure that the Workspace and the associated child Workspaces have been delivered to the parent Workspace also because you cannot rollback the deletion.

2. Click Delete on the Workspace Toolbar.

   A warning message appears, for example, as follows:
   ```
   Are you sure you want to delete the Integration Workspace: <Workspace_name>?
   ```

   > **Note:** If the Workspace is active or configured in the WorkspaceBranchName component parameter, then a message displays that you cannot delete the Workspace.

3. Click Delete Workspace.

Difference Between Deleting Development and Integration Workspace

The difference between deleting a development Workspace and an Integration Workspaces are highlighted in the following table.

Development Workspace	Integration Workspace
Can be deleted by the creator of the Workspace or any user that has the Workspace Administrator responsibility.	Can be deleted only by users that have the Workspace Administrator responsibility. This user must ensure that there is no one actively using the Workspace or any of its children because the deletion is permanent. **Note:** If the Workspace or any of its child Workspaces are configured for migration, then you cannot perform migration once the source Workspace is deleted.
Can be deleted only if the status on the *Workspace Dashboard* is *not* Delivered or Submitted for delivery.	Can be deleted only if the status on the *Workspace Dashboard* is *not* Edit-In-Progress or Rebase-In-Progress. It is not possible to rollback the deletion to recover data. This is an administrative activity.
Can be deleted only if there are no child Workspaces associated with it.	Can be deleted even if there are child Workspaces associated with it.

Using Parallel Development Tools for Workspaces

Developer users can perform the following tasks:

- *Applying Parallel Development Using Workspaces Hierarchy*.
- *Understanding Development Workspaces*.
- *Creating Development Workspaces*.

Applying Parallel Development Using Workspaces Hierarchy

After administrators enable the *parallel development using Workspaces* feature in a working environment, developers can create the N levels of hierarchy for the parallel-development Workspaces. For more information, see *Administering Parallel Development Using Workspaces* and *Enabling the Parallel Development Feature for Workspaces*.

You can apply a parallel-development hierarchy to create Integration Workspaces (IWS). The hierarchy of Workspaces can be in the following combinations:

- IWS, IWS, followed by another IWS.

  This combination is applied to the N levels. You can have multiple Integration or Development Workspaces at one level that are parallel under one parent IWS.
- IWS, Development Workspace, followed by another Development Workspace.

  This combination is applied to the N levels. You can have multiple Development Workspaces at one level that are parallel under one parent (Integration or Developer) Workspace.

Note: You cannot create IWSs using this combination: IWS, Development Workspace, followed by another IWS.

For more information about IWSs, see *Understanding Integration Workspaces*. For more information about the Development Workspaces, see *Understanding Development Workspaces*.

Understanding Development Workspaces

You can create Development Workspaces by branching from an Integration Workspace at the immediate parent level.

After opening a Development Workspace, you can directly edit the Workspace-supported repository object and complete the development configurations.

The names of all Development Workspaces start with the prefix `dev_<userid>_`.

Working on Development Workspaces, note that:

- Administrators can perform the following Workspace operations on Development Workspaces: open, edit Workspaces that support repository objects and complete their development configuration, submit for delivery, and rebase.

- All developers must be able to create Development Workspaces branching from Integration Workspaces. For a new developer, grant the following two responsibilities:
  - **Composer Administrator.** This provides access to the *Workspace Dashboard*.
  - **Web Tools User.** This provides access to the views throughout the Web Tools application, including the Applet List View, Applet Designers, and so on.
- While creating Development Workspaces on a specific release, development members should communicate with the administrators to select an Integration Workspace and define it as the parent Workspace.
- Users with the developer permissions can create the parallel-development Workspaces by selecting one of their parent release Integration Workspaces as the parent Workspace.

  For example, if the parent release Integration Workspace is CRM20xx_apps, the developer's environment displays the parallel-development Workspace trees as the following:

  ```
  crm20xx_apps, dev_sadmin_l1, dev_ccheng_l2, followed by dev_cmorris_l3
  ```
- Development teams can edit repository metadata or change the repository configuration only through the development Workspaces.
- Users with the developer permissions can open the Development Workspaces that they own to edit the repository configuration, checkpoint the version, revert the version changes, rebase the Workspaces, change the state of the Workspaces to Submitted to Delivery, and delete the Workspaces (that is in the editable state if that Workspace does not have a child Workspace).
- Users or developers can preview the Development Workspace in the Siebel application by clicking the Inspect button for the repository changes they made to that version. However, they cannot perform the Inspect task on Integration Workspaces.
- Workspace developers cannot deliver their Workspace changes directly to Integration Workspaces after they complete the repository configurations on their Development Workspaces. In such cases, they must change the status of the Workspaces to Submitted for Delivery before the Siebel administrators approve and perform the delivery process.

Note: Delivering the changes to the parent Development Workspace delivers the changes and also publishes the changes.

Creating Development Workspaces

The following procedure describes how to create a Development Workspace.

To create a Development Workspace

1. In the Workspace Explorer (in the *Workspace Dashboard*), select the parent Integration Workspace or any parent Development Workspace.
2. Click Create on the Workspace Toolbar:

   The Create Workspace window that opens contains the following elements:
   - Name of the selected parent Workspace.
   - Name field.
   - Comment field.

  - Cancel and Create Workspace buttons.

3. Append the Workspace name text to the prefix `dev_ccheng_`.

   For example, `dev_ccheng_bug2468`.
4. (Optional) Enter a comment in the Comment field.
5. Click Create Workspace to create the Development Workspace.

7 Using Siebel Script Editors

Using Siebel Script Editors

This chapter describes how to use the Siebel Script Editors. It includes the following topics:

- *Using the Siebel Script Editor*
- *Using eScript*
- *Using Siebel Server Script Debugger*
- *Using the Script Profiler*

Using the Siebel Script Editor

This topic describes how to use the Siebel Script Editor. It includes the following information:

- *Overview of Using Siebel Script Editor*
- *Opening the Siebel Script Editor*
- *Setting Options for Siebel Script Editor*
- *Validating Script Syntax*
- *Guidelines for Using Siebel Script Editor*
- *Guidelines for Testing a Script*
- *Suppressing Error Messages*
- *Debugger Service*

Overview of Using Siebel Script Editor

Siebel Script Editor is an editor that allows you to create and maintain scripts that use Siebel VB, Siebel eScript, or Browser Script. If you cannot use a declarative configuration to implement the functionality you require, then you can use the following editors to add a script to a Siebel object:

- **Server Script Editor.** Allows you to create and modify Siebel eScript or Siebel VB.
- **Browser Script Editor.** Allows you to create and modify Browser Script that runs in the Siebel application.

> **Note:** Oracle recommends that customers use the custom Presentation Model (PM) or Physical Renderer (PR) mechanism for any new client browser scripts. Also, customers should eventually migrate all existing legacy browser scripts to the PM or PR model. For more information, see *Configuring Siebel Open UI* .

For more information, including a description of scriptable events and callable methods on browser objects, see *Siebel Object Interfaces Reference* , *Siebel eScript Language Reference* , and *Siebel VB Language Reference* .

About the Siebel Script Editor

You can use the Siebel Script Editor to edit Siebel VB script or Siebel eScript script. It allows you to do the following:

- Cut, copy, and paste the text from one location to another location in the Siebel Script Editor. You can also cut and paste text into the Siebel Script Editor.
- Import or export a Siebel script.
- Associate a given Siebel script with a predefined object event, such as a PreSetFieldValue event for a business component. For more information, see *About Predefined Objects*.
- Debug a custom script. For more information, see *Using Siebel Server Script Debugger*.
- Compile a custom script. For more information on how to deliver Workspaces, see *Delivering Workspaces*.

The Siebel Script Editor uses a window that is similar to the Windows Notepad editor. It includes the following elements:

- Title bar.
- Drop-down list that you can use to specify the application configuration content.
- Toolbar containing the following buttons: Activate, Inactivate, Back, Check Syntax, Settings, and a drop-down list containing all applications. For more information about each of these options, see *Siebel Script Editor Toolbar*.
- Text entry window, which is disabled if the Workspace is not editable.
- Vertical and horizontal scroll bars that you can use to navigate.

Object Types You Can Script

You can use the Siebel Script Editor to add a script to the following object types:

- Application
- Applet
- Business Component
- Business Service

An object definition for each of these object types includes a Scripted property. If this property includes a check mark, then this object includes a script.

Siebel Script Editor Toolbar

The following table describes the buttons on the Script Editor toolbar. The button name appears when you hover the cursor over each button.

Note: Web Tools implicitly saves your script.

Button	Name	Description
📜	Activate	Click to change a script from inactive to active. After you activate a script in the script editor, the applicable Script Editor tree for the same Type and Object is automatically updated. The Activate icon is active or enabled (that is, you can click it) when the Workspace is edit-in-progress and the script is inactive. The Activate icon is disabled (that is, you cannot click it) under the following conditions:

Button	Name	Description
		• When the script has no data. • When the current selected node is a folder. • When the Workspace is read-only and the script is active or inactive. • When the Workspace is edit-in-progress and the script is active.
📜	Inactivate	Click to change a script from active to inactive. After you inactivate a script in the script editor, the applicable Script Editor tree for the same Type and Object is automatically updated. The Inactivate icon is active or enabled (that is, you can click it) when the Workspace is edit-in-progress and the script is active. The Inactivate icon is disabled (that is, you cannot click it) under the following conditions: • When the script has no data. • When the current selected node is a folder. • When the Workspace is read-only and the script is active or inactive. • When the Workspace is edit-in-progress and the script is inactive.
↩	Back	Closes the Script Editor and returns to the view you came from.
☑	Check Syntax	Validates the syntax of the selected script.
⚙	Settings	Opens the Settings drop-down menu. This menu contains the following options, which you can configure for the Script Editor and script engine: • **Formatting options.** You can choose any of the following: Auto-Close Brackets, Display Line Numbers, Wrap Long Lines, Highlight Active Line, Match Brackets, Indent with Tabs. • **Script Assist.** You can choose any of the following: Enable Method Listing, Enable Auto Complete, Enable Favorites. • **Engine Settings.** You can choose any of the following: Fix And Go, Default Login, Enable Warnings, Deduce Types. • **Languages.** You can choose the following: Default language for new scripts (eScript or Visual Basic). For more information about the Settings you can configure for the Script Editor, see *Setting Options for eScript* and *Setting Options for Siebel Script Editor*.

Button	Name	Description
📜	This icon shows a script with code and appears adjacent to the built-in event handlers, such as, BusComp_PreWriteRecord.	The ability to edit code depends on whether the current open Workspace is editable or not.
▯	This icon shows an empty script (without code) and appears adjacent to the built-in event handlers, such as, BusComp_PreWriteRecord.	The ability to add code depends on whether the current open Workspace is editable or not.

Opening the Siebel Script Editor

This topic describes how to open Siebel Script Editor.

To open Siebel Script Editor

1. In the Object Explorer, select one of the following object types:

    - Application
    - Applet
    - Business Component
    - Business Service

    For example, select Business Component.
2. In the Business Component list, locate and select the object you want to modify.

    For example, select Account.
3. Click Menu (the cogwheel icon) and then select either Edit Server Scripts or Edit Browser Scripts.

    The Siebel Script Editor opens where you can edit, in this example, the server scripts for the Account business component.

Setting Options for Siebel Script Editor

You can set options for working in the Siebel Script Editor, including setting a default scripting language and defining options for debugging.

To set options for Siebel Script Editor

1. Follow the instructions in *Opening the Siebel Script Editor* to open the Siebel Script Editor.

2. Click Settings (the cogwheel icon) on the Script Editor toolbar to configure the following options.

   - **Formatting options.** Select the check box next to any of the following formatting options to enable the option: Auto-Close Brackets, Display Line Numbers, Wrap Long Lines, Highlight Active Line, Match Brackets, Indent with Tabs.
   - **Script Assist.** Select the check box next to any of the following options to enable the option: Enable Method Listing, Enable Auto Complete, Enable Favorites.
   - **Engine Settings.** Select the check box next to any of the following options to enable the option: Fix And Go, Default Login, Enable Warnings, Deduce Types.
   - **Languages.** Select the check box next to the following option and then either eScript or Visual Basic from the drop-down list: Default language for new scripts (eScript or Visual Basic).

   For more information, see *Siebel Script Editor Toolbar* and *Development Options for Scripting*.

Validating Script Syntax

Siebel Script Editor includes a syntax checker that you can use to make sure your script compiles properly.

CAUTION: The Check Syntax feature identifies only syntax errors and errors that occur if an object or variable is not initialized. It does not identify other types of errors and it cannot trap an error that might cause a run-time error. It examines only script that is attached to the current active object. If an error exists in another script, then you cannot compile the repository.

To validate script syntax

1. Open Siebel Script Editor and click Check Syntax on the Script Editor toolbar.

   Web Tools does a test compile and then does one of the following:

   - **If it finds no error,** a message is returned as follows: Check Syntax is completed without errors.
   - **If it finds an error,** a dialog box opens describing the error.
2. If the script includes an error, then click Go to Line in the dialog box.

   Web Tools positions the cursor on the script line that caused the error and highlights the line.
3. Correct the code and repeat Step 1.

   If the syntax of the line that you modified is correct, then the dialog box displays the next error it finds, if any.

   Note: You can modify a script only if the Workspace is editable.
4. Repeat Step 1 and Step 2 until you correct all errors.
5. To commit the changes, move out of Edit view or press `ctrl+s` to save.

Guidelines for Using Siebel Script Editor

If you use the Siebel script editor, then it is recommended that you adhere to the following guidelines:

- To verify the format of your Siebel VB or Siebel eScript script, click Check Syntax on the Script Editor toolbar. The Siebel Compiler displays any format errors it finds and indicates the lines where these errors occur.

  Note: Web Tools implicitly saves changes to your script.

- When pasting text into the Script Editor, avoid using two code blocks that use the same name. To do this, do the following depending on the type of script you use:
  - **Siebel eScript.** Place the code between `function Name {}`. You must enclose all function code within braces, { }.
  - **Siebel VB.** Place the code between `Sub Name` and `End Sub` or within a function, for example, as follows:

    ```
    Function myFunction(...) as X
     // insert code here
    End Function

    Sub mySub(...)
     // insert code here
    End Sub
    ```

- To test your changes, see *Guidelines for Testing a Script*.

Guidelines for Testing a Script

If you test a script, then it is recommended that you adhere to the following guidelines:

- Before using the Debugger Service, set the Web Tools Object Manager parameters required for starting the debugger service, create a subsystem, and then configure its parameters. For more information, see *Debugger Service*.
- To test a script without using the Debugger Service - see the following procedure.
- To test a script using the Debugger Service - see the following procedure.

To test a script without using the Debugger Service

1. Click Check Syntax on *Siebel Script Editor toolbar* to make sure there are no errors in your script.
2. Log in to the Siebel application and go to the Workspace Dashboard.
3. Open the Workspace containing the scripts you implemented, click Inspect, and then test the scenario(s) that will trigger the scripts.

To test a script using the Debugger Service

1. Click Toggle Script Debugger (the bug icon) on the application banner or select the Toggle Script Debugger option from the Debug menu.

   The Server Script Debugger window opens with the usual areas for Watch, Call Stack, and so on.
2. Click Start on the Server Script Debugger toolbar to start the Siebel application in a new browser tab.

For more information about the Server Script Debugger toolbar, see *Accessing Siebel Server Script Debugger*.

3. Log in to the application.

   Siebel CRM runs with the new modifications incorporated.

Suppressing Error Messages

This topic describes how to configure Web Tools to not display the SBL-EXL-00151 error code and error text in the pop-up error message that the RaiseErrorText method creates.

To suppress error messages

1. In Web Tools, select System Preferences from the Tools menu.
2. In the System Preferences list, set the Suppress Scripting Error Code system preference value to TRUE, as shown in the following table.

System Preference Name	System Preference Value
Suppress Scripting Error Code	TRUE

Debugger Service

The Debugger Service handles communications between Web Tools and an Application Object Manager (AOM), such as Siebel Call Center, when debugging scripts or Workflow Processes. It can be run on any or all Siebel Servers in the Design Repository (DR) development environment. The service is defined as a Java process (DBGService.jar).

In general, when you start a debugging session, a Siebel Server will attempt to connect to a Debugger Service running on the same machine as the AOM. If there is no Debugger Service available on that machine, it will attempt to connect to a designated primary host.

Configuring the Primary Host

To configure the primary Debugger Service, you must perform a one-time setup task to create a subsystem named DBGSVC using the subsystem type "DbgProcCfg" and then set the following parameters:

- **Host.** The hostname or server where the debugger service is hosted in an enterprise. It is the default host/server that the Debugger Service is going to connect to if there is no debugger service running on the same host/server that the Web Tools component is running on.
- **Port.** The port at which the debugger service is run as a socket server. The value of this parameter must be between 0 and 65535 and should not be a well-known port or a port in use by any other process on that host. The configured value is passed as a parameter to the AOM from the Web Tools session, allowing the AOM to

communicate back to the debugger service on that port. For information on common well-known ports, see *https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml*.

> **Note:** If the Application Interface is running outside the firewall, then this port must be open to the Siebel Server machine where the debugger service runs.

- **TxnTimeout.** This is an internal parameter used by the debugger service for reading the data at any given time interval. The value of this parameter should not exceed 10 digits, and the recommended value is 50 or 100 milliseconds. The configured value is passed as a parameter to the AOM.
- **MaxTasks.** The maximum number of debugger sessions that the debugger service can run at a time. The value of this parameter should not exceed 10 digits, and the recommended value is 100. The configured value is passed as a parameter to the debugger service.

The Web Tools Script Debugger will not connect to the Debugger Service (DBGService.jar) if:

- The named DBGSVC subsystem is not created
- Any of the subsystem parameters are not set

Creating a Subsystem and Configuring Parameters from the Server Manager Utility

The following commands show how to create a named subsystem, list its parameters, and then configure its parameters from the Server Manager command-line utility (srvrmgr):

- To create a named subsystem, go to server manager and type the following command:
  ```
  create named subsystem DBGSVC for subsystem DbgProcCfg
  ```
- To list the parameters, type the following command:
  ```
  list parameters for named subsystem DBGSVC
  ```
- To configure or change the values of parameters, type the following commands:
  ```
  change parameter Host=<machine_name> for named subsystem DBGSVC
  change parameter Port=<port> for named subsystem DBGSVC
  change parameter MaxTasks=<tasks> for named subsystem DBGSVC
  change parameter TxnTimeout=<txn_timeout> for named subsystem DBGSVC
  ```

  For example:

  ```
  change parameter Host=MySiebelServer.domain.com for named subsystem DBGSVC
  change parameter Port=10000 for named subsystem DBGSVC
  change parameter MaxTasks=100 for named subsystem DBGSVC
  change parameter TxnTimeout=50 for named subsystem DBGSVC
  ```

Configuring the Debugger Service Properties

In addition to specifying the parameters in the Server Manager, the Script Debugger Java service must be configured in the dbgsrvr.properties file, located in the `.../ses/siebsrvr/CLASSES` folder. Where they have the same property names, values must match the subsystem parameter values defined previously:

- **Port.** This parameter is described in *Configuring the Primary Host*.
- **TxnTimeout.** This parameter is described in *Configuring the Primary Host*.
- **MaxTasks.** This parameter is described in *Configuring the Primary Host*.

- **ShutDownPort**. The port at which the debugger service is waiting for a stop command to exit gracefully. The value of this parameter must be between 0 and 65535 and should not be a well-known port or a port in use by any other process on that host.

Configuring Additional Instances of the Debugger Service

If using multiple instances of the Debugger Service, then note the following:

- You can start the Debugger Service on any (or all) Siebel Servers where you want to host the additional Debugger Service.
- You must configure the dbgsrvr.properties file on all Siebel Servers where the Debugger Service will be hosted. It is recommended that all values be the same on all hosts. Port is required to be the same.
- The Web Tools Object Manager component will check whether the Debugger Service is running or not on its respective server:
  - If the Debugger Service is running, then that instance will be used.
  - If the Debugger Service is not running, then the primary Debugger Service defined by the DBGSVC named subsystem Host parameter will be used.

Enabling the Debugger Service

A Windows Service to launch the Debugger Service is created during installation of the Siebel Server. If it has been removed or the installer failed to create it, the service can be recreated by step 1 in the following procedure.

On UNIX platforms, the Debugger Service must be launched as a daemon process.

To enable the Debugger Service on Microsoft Windows:

1. If necessary, run the InstallDbgservice.bat batch file (located under `.../ses/siebsrvr/bin`) to create the DbgService entry in the standard Windows Services Control Panel. This is a one-time task.
2. On all hosts where the Debugger Service should run, configure the DbgService entry to start with the "Automatically (Delayed)" setting in the Services Control Panel. Using the "Delayed" option allows the Siebel Server to start before the Debugger Service.

   Note: You can also manually control the DbgService (the debugger service) as required in the Windows Services Control Panel using the standard Start, Pause, Stop, and Restart buttons.

To enable the Debugger Service on UNIX:

- Edit the existing Siebel Server startup and shutdown scripts as follows:
  - Run `./startdbgsvc.sh` (located under `.../ses/siebsrvr/bin`) to start the debugger service
  - Run `./stopdbgsvc.sh` (located under `.../ses/siebsrvr/bin`) to stop the debugger service

Using the ST eScript Engine

This topic describes how to use Siebel eScript. It includes the following information:

- *Overview of Using eScript*
- *Setting Options for eScript*

- *Using Fix And Go*
- *About Using Script Assist*
- *Using Script Libraries with eScript*
- *Using Running Tool Tip*

Overview of Using the ST eScript Engine

Siebel eScript is compliant with the ECMAScript Edition 4 standard. ECMAScript is the implementation of JavaScript as defined by the ECMA -262 standard. The Siebel eScript includes the following capabilities:

- **Performance.** Provides higher throughput with a lower CPU and memory footprint in situations where you use a significant amount of script. The result is improved performance and lower maintenance on heavily scripted events.
- **Scalability.** Superior performance even if many users concurrently run scripts.
- **Strong typing.** Supports strong typing that is compliant with the ECMAScript Edition 4 standard. Strongly typed objects result in scripts that are more functional and improved performance.
- **Functionality.** Adds new functionality, such as Script Assist, script libraries, favorites, and Fix And Go. For more information, see *About Using Script Assist* and *Using Fix And Go*.

For more information, see *Siebel eScript Language Reference* .

Setting Options for the ST eScript Engine

This topic describes how to set options for the ST eScript Engine.

To set options for the eScript

1. From any supported object in Web Tools, click Menu (the cogwheel icon) and then select Edit Server Scripts to open Siebel Script Editor.
2. In the Script Editor, click Settings on the Script Editor toolbar to configure the settings described in the following table. It is recommended that you enable all of these options.

Setting	Description
Fix And Go	If this check box is selected, then Web Tools allows you to modify the server script (provided the Workspace is editable) during the debugging process, and then debug and test the script before delivering the server script into the MAIN Workspace. For more information, see *Using Fix And Go*.
Default Login	This option applies only during the Fix And go process. If this check box is selected, then Web Tools allows automatic login to the application being debugged in the Server Script Debugger. For more information, see *Using Fix And Go*.
Enable Warnings	If this check box is selected (that is, includes a check mark), then Web Tools displays script compile warning messages. For information, see *About Siebel eScript Warnings*.

Setting	Description
Deduce Types	If this check box is selected, then Web Tools infers the type of local variables. For more information, see *How Siebel eScript Determines the Type of Variables That a Script Uses*.

For more information about all the settings you can configure for the Script Editor, see *Siebel Script Editor Toolbar*.

About ST eScript Engine Warnings

The Siebel eScript displays warnings that notify you of the following potential problems that might occur if you deliver your custom script:

- Referencing a method or property that is not predefined
- Referencing an undeclared identifier
- Using a variable before it initializes this variable
- Using a redundant declaration of an untyped variable
- Calling a function that includes an insufficient number of arguments

These errors might cause a run-time failure. You can use these warnings to help fix errors before you deliver your script. To enable or disable the Enable Warnings option, see *Setting Options for eScript* .

Example of a Warning Message

The following code is an example of a compile warning message that Web Tools creates after a run-time failure:

```
function foo(a)
 {
 var oApp: Application;
 oApp.myMethod ();
 return;
 }

foo ();

Semantic Warning around line 5:Variable oApp might not be initialized.
Semantic Warning around line 5:No such method myMethod
Semantic Warning around line 10:Calling function foo with insufficient number of arguments.
Unhandled Exception: Function expected
```

How the ST eScript Engine Determines the Type of Variables That a Script Uses

Type deduction is a feature of the Siebel eScript that determines the type of local variables that a script uses. It scans the assignments that the script makes to each local variable. If the Siebel eScript can successfully determine the type for all local variables, then the compiler performs strict type checks and creates statically bound code that runs faster and uses less memory. This configuration might introduce more compile warnings as a result of performing these type checks. It cannot determine the type in all situations. It is recommended that you type your script. To enable or disable type deduction, see *Setting Options for eScript* .

Example of a Script That Deduces the Local Variable Type

The following script deduces the type of the oDate local variable to the Date. It then creates a warning about the MyMethod method. This method is not defined. This script fails at run time:

```
function goo()
```

```
     {
     var oDate;
     oDate = new Date ()
     oDate.myMethod ();
     return;
     }

    goo ()

    Semantic Warning around line 19:No such method myMethod
    Unhandled Exception: 'myMethod' is not defined
```

Using Fix And Go

Fix And Go allows you to edit a script in a Web Tools session and then test your modifications in the Siebel application without closing the client or recompiling the script. This feature saves you time in script development, testing, and debugging. You can use Fix And Go only with server script.

To use Fix And Go

1. In the *Workspace Dashboard*, open the Workspace that you want.
2. Click Toggle Script Debugger (the bug icon) on the application banner or select Toggle Script Debugger from the Debug menu to open the Server Script Debugger window.
3. Click Start on the Server Script Debugger toolbar to start the application in debug mode.

   For more information about the Server Script Debugger toolbar, see *Accessing Siebel Server Script Debugger*.
4. Log in to the application, navigate to the view which uses the server side script and verify that the server script is working as it should be.
5. Now in Web Tools, change the server script in the Script Editor and save the changes.

   Notice that this enables the Restart button on the Server Script Debugger toolbar.
6. Click Restart on the Server Script Debugger toolbar.

   This restarts the target application, navigating the user to either the login page (if Default Login is False) or the view in Step 4 (if Default Login is True).

   > **Note:** Default Login works with employee facing applications (like Call Center) and without any Single Sign-On configuration or custom login page. Default Login expects that the AllowAnonUsers component parameter is set to FALSE for components.

   For more information about all the settings you can configure for the Script Editor, see *Setting Options for eScript* and *Siebel Script Editor Toolbar*.

The following procedure shows how to enable (and disable) the Fix And Go feature. Fix And Go is enabled by default in Siebel CRM.

To enable/disable Fix And Go functionality

1. Follow the instructions in *Opening the Siebel Script Editor* to open the script editor.
2. Click Settings on the Script Editor toolbar.
3. Under Engine Settings:
   - Select the check box next to Fix And Go to enable the feature.

- Deselect the check box next to Fix And Go to disable the feature

About Using Script Assist

Script Assist is part of the Script Editor. To help you develop a script, it inspects object definitions, and then makes information about these object definitions available to you. It includes the following functionality:

- **Syntax highlight.** Uses color to highlight reserved words, data types, operators, and other syntax in Siebel VB and Siebel eScript script. You cannot modify these colors. The following table lists the colors that it uses and the item of code to which each color corresponds.

Item In the Code	Color Name	Hex Color Code
Reserved word or keyword	Dark Purple	#708
String literal	Dark Red	#a11
Number literal	Dark Green	#164
RegExp literal	Orange	#f50
Comment	Brown	#219
Global Property	Ultramarine-1	#219
Function. For Siebel VB only	Ultramarine-2	#30a
Declaration for Siebel eScript	Blue	#00f
Declaration for Siebel VB	Black	#000
Variable for Siebel eScript	Cobalt Blue	#05a
Variable for Siebel VB	Black	#000

- **Method list.** Displays a list of methods and properties that are available for an object. For more information, see *Icons that the Script Assist Window Contains*.
- **Repository inspection.** Inspects objects and object types in the repository without requiring you to type a string literal. This functionality results in fewer mistakes in your script. It also understands predefined constants for a business component method.

- **Favorites.** Uses italics to indicate the most frequently used object, method, or property name in the Script Assist window. Favorites exist for only a single Web Tools session. When you log out of Web Tools, it clears these favorites. If you create a new function, then you must add it to the declarations, and then save the script modifications. If you do not do this, then Web Tools does not display the function as a favorite.
- **Script libraries.** Allows you to call a business service function after you declare the business service. You do not declare property sets or make an InvokeMethod call. A script library helps you develop code that is reusable and modular. For more information, see *Using Script Libraries with eScript*.
- **Auto complete.** Automatically completes an entry when you enter a minimum number of unique characters in the Script Assist window and press `Ctrl + Space`. For example, if you enter `The` and press `Ctrl+Space`, then Web Tools automatically enters the word `TheApplication`.
- **Auto indent.** Maintains a running indent. If you press the Return key or the Enter key, then it inserts spaces and tabs that left-justifies the code.
- **Tool tips.** Allows you to view descriptions of the method arguments that you use.
- **Includes child scripts.** Script Assist can parse a script that you write on a business component, applet, or business service. You can use the Application drop-down list to choose the Siebel application that includes the child script. Script Assist displays the scripts that are written on this application object in the Script Assist window.
- **Includes scripts you write in the general section.** A script that you write in the general section of the script explorer window is available in the Script Assist window. For example, if you write a helper function named Helper in the general section of the current script, and if you start Script Assist, then it includes Helper in a pop-up window.

Icons that the Script Assist Window Contains

The following table describes the buttons (methods and properties) that the Script Assist window shows when you choose an object. For more information, see *Opening the Script Assist Window*.

Button	Name	Functionality
📜	Object Built-in Function	Object Built-in Function
📜	Object Class	Object Class
📜	Object Method	Object Method
📜	Object Primitive	Object Primitive

Button	Name	Functionality
	Object Property	Object Property
	Object Type	Object Type

Opening the Script Assist Window

You must enable the following Script Assist options in the Script Editor settings before opening the Script Assist window:

- Enable Method Listing
- Enable Auto Complete

For more information about enabling these options, see *Setting Options for Siebel Script Editor*.

To open the Script Assist window

1. In Siebel Script Editor, type the name of the object that you want followed by a period (`.`).

   Web Tools returns the methods that are available for the specified object in a drop-down list.
2. Choose the required method that you want from the list to add to the script.

Or

1. In Siebel Script Editor, type the first few characters of the required object name and then press `Ctrl+Space`.
2. Web Tools automatically returns the object name (or names) containing the characters to add to the script.

Using Script Assist with a Custom Siebel eScript Method

If a script references a custom Siebel eScript method on a business component, and if this script does not reside on the same business component that includes this custom method, then Script Assist does not recognize the custom method and cannot display data from it. For example, you create a business component method named MyMethod on the Account business component. You then create a script on the Contact business component that references the MyMethod method. In this example, Script Assist does not include any information about your custom My Method method.

Using Script Libraries with the ST eScript Engine

A *script library* is a part of the Siebel eScript that allows you to call a business service method from a script. This topic describes how to make a custom business service method available to a business service script library and how to call this method in the script library. Using a script library is optional. Siebel CRM supports code you write before Siebel CRM version 8.0. For more information about script libraries, see *Siebel eScript Language Reference* .

Making a Custom Business Service Method Available in a Script Library

This procedure describes how to make a custom business service method available in a script library.

To make a custom business service method available in a script library

1. Create a script for a business service method.
2. Validate the script - for more information, see *Validating Script Syntax*.
3. Make sure to select the External Use property of the business service (make sure it contains a check mark).

   Web Tools adds the custom business service method to the script library.

Using a Script Library to Call a Custom Business Service Method

After you make a business service available for use in Web Tools, the following applies:

- You can call a business service method of this business service from another script.
- The Script Assist window displays these business service methods. For more information, see *About Using Script Assist*.

To use a script library to call a custom business service method

1. Make sure the following options are enabled in ScriptAssist:

   - Enable Method Listing
   - Enable Auto Complete

   For more information about setting these options, see *Setting Options for Siebel Script Editor*.
2. In the Siebel Script Editor, type the name of a business service object followed by a period (.).

   Web Tools displays the business service methods that are available for the business service.
3. Choose the business service method that you want to add to the script.
4. Validate the script - for more information see, *Validating Script Syntax*.

Example of Using a Script Library

In the following example, the mathService business service is enabled for use in Web Tools. It uses a business service method named square:

```
function square (x)
{
 return (x * x);
}
```

If you use a script library to call an argument, then the compiler examines the argument types to make sure the argument that your script calls is valid and is compatible. The following code calls a business service method named square:

```
var oBS: Service = TheApplication().GetService ("mathService");
var value = 10;
var square_value = oBS.square (value);
```

Using Running Tool Tip

Running Tool Tip is a help feature in the Siebel Script Editor that you can use to write a script. If you enter a method name in the Running Tool Tip window, then it displays the method arguments that you can use with this method. If you enable Script Assist, then Running Tool Tip is enabled by default. For more information, see *About Using Script Assist*.

To use Running Tool Tip

1. Open the Siebel Script Editor - for more information, see *Opening the Siebel Script Editor*.
2. In the Script Editor window, enter a method name followed by an open parenthesis.

   The Running Tool Tip window opens. This window displays the method name and the method arguments. It uses italicized text to suggest an argument. This window disappears (is hidden) after you enter all the required method arguments.

How Running Tool Tip Differs from Tool Tips in Script Assist

If you enter a function name followed by an open parenthesis in the Script Editor, then Script Assist or Running Tool Tip opens depending on the type of function you enter.

Running Tool Tip

If you enter a simple call expression, then the Running Tool Tip window opens. A call expression allows you to send anything to the function according to the type of argument. For example, if you use the following code to enter a data object function, then Siebel CRM can send a value that you choose to this function:

```
Date.SetFullYear
```

If you enter the following:

```
var oDate = new Date();
oDate.SetFullYear(
```

Then the following code appears in the Running Tool Tip window.

```
oDate.SetFullYear(year, month, date)
```

Script Assist

If you enter a collection function, then the Script Assist window opens in Web Tools. This window includes a list of methods and properties that you can choose for an object. A *collection function* is a type of function that includes a finite set of values. You can send values to a collection function. For example, if you enter the following code, then the fields that are defined only for this business component are displayed:

```
BusComp.GetFieldValue
```

The following code is an example:

```
var bo = TheApplication().GetBusObject(
```

For more information about using functions and expressions in Siebel eScript, see *Siebel eScript Language Reference* .

Using Siebel Server Script Debugger

This topic describes how to use the Siebel Server Script Debugger. It includes the following information:

- *Overview of Using Siebel Server Script Debugger*
- *Accessing Siebel Server Script Debugger*
- *Setting Debug Options for Siebel Server Script Debugger*
- *Using Breakpoints*
- *Using the Call Stack Window*
- *Using the Watch Window*
- *Tracing a Script*

Overview of Using Siebel Server Script Debugger

Siebel Server Script Debugger helps you debug a script and identify any errors in a script written in Siebel VB or Siebel eScript. It displays variables and their values in the Script Editor window and a diagnostic window. You can use Server Script Debugger to slow down or suspend a script from running so that you can monitor the logical flow of your script and examine the contents of variables. Siebel Server Script Debugger allows you to do the following:

- **Set a breakpoint.** A breakpoint is a marker in a line of code that stops code from running at the line marker, thereby allowing you to examine the state of the program at that line. For more information, see *Using Breakpoints*.
- **Step over a line of code.** That is, skip a given line of code. For example, if the current code line contains a subroutine or function, then the debugger executes the subroutine or function and displays the result without debugging each line in that subroutine or function.
- **Step into a subroutine.** Runs one line of code according to the following:
  - **The current line calls a subroutine or function.** The debugger stops at the first line of the subroutine or function that the current line calls.
  - **The current line does not call a subroutine or function.** The debugger stops at the next code line.
- **Examine the value of a variable.** The debugger includes a Watch window that displays variables and their values. You can examine these values while a script runs. For more information, see *Using the Watch Window*.

Accessing Siebel Server Script Debugger

The following procedure shows how to access Siebel Server Script Debugger.

To access Siebel Server Script Debugger

- Do one of the following in Web Tools:
  - Click Toggle Script Debugger (the bug icon) on the application banner.
  - Select Toggle Script Debugger from the Debug menu.

The Server Script Debugger window opens.

The following table describes the buttons on the Server Script Debugger toolbar. The name of each button appears when you hover the cursor over the button.

Button	Name	Description
▶	Start or Continue	Click to start debugging the Siebel application based on specified settings, or to continue with a debugging session that was previously started but then stopped at a breakpoint.
⏸	Break	Click to stop the script that is currently running. If Siebel VB or Siebel eScript is not running, then Web Tools does nothing.
⏹	End	Click to stop the Siebel application and return to the Siebel Script Editor window.
↺	Restart	Click to trigger a relaunch of the application during the Fix And Go process.
↳	Step Into	Click to run the next line of a script. If this line calls a subroutine or procedure, then Web Tools runs this subroutine or procedure.
↷	Step Over	Click to run the first line of script that occurs after the current subroutine or procedure. Web Tools continues to run this script in the current subroutine or procedure.
↪	Step To Cursor	Click to run all lines of code up to the line that includes the cursor.
◉	Toggle Breakpoint	Click to set or remove a breakpoint on a line of code. For more information about breakpoints, see *Using Breakpoints*.
⊘	Clear All Breakpoints	Click to remove all breakpoints from all scripts.
⚙	Settings	Click to open the Debugging Settings dialog. For more information about debug settings, see *Setting Debug Options for Siebel Server Script Debugger*.
↗	Undock	Click to undock the Server Script Debugger window so that you can move, resize, and/or position it anywhere on your screen.
↙	Dock	Click to dock the Server Script Debugger window under the Script Editor window.
✕	Close	Click to close the Server Script Debugger window.

Setting Debug Options for Siebel Server Script Debugger

The following procedure shows how to configure the debug settings for Siebel Server Script Debugger.

To set debug options

1. In Web Tools, do one of the following to open the Server Script Debugger window:
   - Click Toggle Script Debugger (the bug icon) on the application banner.
   - Select Toggle Script Debugger from the Debug menu.
2. On the Server Script Debugger toolbar, click Settings to open the Debugging Settings dialog.

   For more information about the Server Script Debugger toolbar, see *Accessing Siebel Server Script Debugger*.

   The settings available in the Debugging Settings dialog are described in the following table.

Setting	Description
Adjust breakpoint to next valid line	Select this check box if you want Web Tools to create a breakpoint at the next valid code line when a breakpoint is deleted from an invalid code line.
Always enter the debugger when an error occurs.	Select this check box if you want Web Tools to automatically open the Server Script Debugger window when a script error occurs.
Make debugger window active when debugging	Select this check box if you want Web Tools to always show the Server Script Debugger window when it is in debug mode.
Application <callcenter (enu)>	This is a drop-down list where you must select the application to use for debugging. The application that you select here must be running on the same node as the Web Tools client where the debugger service is running. This is because the application URL will be constructed using the same base as the Web Tools URL.

   For more information, see *Development Options for Scripting*.

Using Breakpoints

A *breakpoint* is a marker in a line of code that stops code from running at the line marker, thereby allowing you to examine the state of the program at that line.

To use breakpoints

- Do one of the following in Siebel Server Script Debugger:

- Place the cursor on the line of code where you want to set a breakpoint, and then click Toggle Breakpoint on the Server Script Debugger toolbar. For more information about the Server Script Debugger toolbar, see *Accessing Siebel Server Script Debugger*.
- Place the cursor on the line of code where you want to set a breakpoint, and then select Toggle Breakpoint from the Debug menu.
- Click the gutter area adjacent to the line of code where you want to set a breakpoint to turn on the breakpoint.
- Position the cursor in the gutter area adjacent to the line of code where you want to set a breakpoint, right-click and then select Toggle Breakpoint from the popup menu that appears.

If a breakpoint already exists on the line of code, then the breakpoint is removed. If a breakpoint does not already exist on the line of code, then a breakpoint is added.

Using the Call Stack Window

The Call Stack window includes a list of subroutine and function calls that Web Tools runs prior to the current code line.

To use the Call Stack window

1. Open Siebel Server Script Debugger - for more information, see *Accessing Siebel Server Script Debugger*.
2. Start the Siebel application from the Server Script Debugger window by clicking Start on the Server Script Debugger toolbar.
3. In the Server Script Debugger window, click Call Stack while running the debugger.

   Each line in the Call Stack window that opens represents a subroutine that Web Tools entered but did not complete. If you select an entry in the list, then the following occurs:

   - The Server Script Debugger window displays the line of code that makes the call.
   - The Variable window displays the variables that are associated with the procedure that makes the call.

Using the Watch Window

The Watch window displays variables and variable values. You can use it to monitor these values while a script or Workflow Process runs. The Watch window supports the following variable types:

- Local variables
- Global variables
- Shared global variables
- Application variables
- Persistent profile properties
- Dynamic profile properties

For a detailed example that uses the Watch window, see the information about defining a Workflow Process that closes obsolete service requests in *Siebel Business Process Framework: Workflow Guide* .

To use the Watch window

1. Write a script to an object.
2. Start the Siebel application from the Server Script Debugger window by clicking Start on the Server Script Debugger toolbar.

   For more information about the Server Script Debugger toolbar, see *Accessing Siebel Server Script Debugger*.
3. In the Server Script Debugger window, click Watch while running the debugger.

   You can monitor values while the script runs in the Watch window that opens.

Tracing a Script

You can run a trace on an allocation, event, or SQL command. You can start a trace for a user account, such as your development team. The Siebel Server saves the trace information to a log file. Tracing a script is not the same as tracing a file. For more information about tracing a file, see the information that describe the Trace, TraceOn, and TraceOff methods in *Siebel Object Interfaces Reference* .

To trace a script

1. Log in to the Siebel application, navigate to the Administration - Server Management screen, and then the Components view.
2. Choose the component that you want to trace.
3. Navigate to the Component Event Configuration view.
4. Locate the Object Manager Extension Language Log event.

   If this event does not exist, then the component you chose in Step 2 does not support logging. Most components support logging, but some do not. If necessary, repeat Step 2 but choose another component.
5. Set the Log Level to 1.

   To disable logging when you are done, you can set the Log Level to 0 (zero).
6. Navigate to the Component Parameters view.
7. (Optional) To display only the script tracing parameters, enter a query using the values from the following table.

Field	Value
Parameter Alias	Trace*
Subsystem	Object Manager

8. Set one or more tracing parameters using the values from the following table.

Object to Trace	Alias	Description
Allocations	TraceAlloc	Enter `0` (zero) to disable logging. Enter `1` to enable logging.

Object to Trace	Alias	Description
Events	TraceEvents	Enter `0` (zero) to disable logging. Enter `1` to enable logging.
SQL Commands	TraceSql	Enter `0` (zero) to disable logging. Enter `1` to enable logging.
Users	TraceUser	Enter a comma-separated list of user names. Do not use spaces. For example, you can enter the following: `sadmin,mmasters,hkim,cconnors`

To set tracing parameters, enter a value depending on when the modification must take affect:

- **Immediately.** You modify values in the Current Value column.
- **After a restart.** You modify values in the Value on Restart column.

Example of a Trace

The following code is an example of a trace that Web Tools creates after the Business Service Simulator runs:

```
ObjMgrExtLangLog ObjMgrExtLangLog 0 00000080475f1578:0 2007-12-12 07:33:45 [User: ] SQLBIND, 90, 1, Y.
ObjMgrExtLangLog ObjMgrExtLangLog 0 00000080475f1578:0 2007-12-12 07:33:45 [User: ] COMPILE, END,
 <unknown>.
ObjMgrExtLangLog ObjMgrExtLangLog 0 00000080475f1578:0 2007-12-12 07:33:45 [User: ] EVENT, BEGIN, Service
[sam], Service_PreInvokeMethod. ObjMgrExtLangLog ObjMgrExtLangLog 0 00000080475f1578:0 2007-12-12 07:33:45
 [User: ] EVENT, END, Service [sam], Service_PreInvokeMethod. ObjMgrExtLangLog ObjMgrExtLangLog 0
 00000080475f1578:0 2007-12-12 07:34:39 [User: ] SQLSTMT, 91,
```

Example of a Detailed Trace

The following code is an example of a trace that Web Tools creates if the log level is set to high. For brevity, only part of this log file is included:

```
2021 2007-12-12 07:46:29 2007-12-12 07:56:46 -0700 000003fd 001 003f 0001 09 SCCObjMgr_enu 11534365 5452
780 d:\21022\ses\siebsrvr\log\SCCObjMgr_enu_0011_11534365.log 8.1 [21022] ENU

ObjMgrLog Info 3 000000b9475f1578:0 2007-12-12 07:46:29 (modpref.cpp (949)) SBL-DAT-50803:
 SharedFileReader:
D:\fs\userpref\SADMIN&Siebel Universal Agent.spf_SDCHS21N625_5452_780: File succesfully opened.

ObjMgrLog Info 3 000000b9475f1578:0 2007-12-12 07:46:29 (modpref.cpp (949)) SBL-DAT-50804: LoadPreferences:
D:\fs\userpref\SADMIN&Siebel Universal Agent.spf: Preferences succesfully loaded.

ObjMgrExtLangLog ObjMgrExtLangLog 0 000000b9475f1578:0 2007-12-12 07:46:29 [User: ] SQLSTMT, 1, SELECT,
 SELECT

T1.CONFLICT_ID,

T1.DB_LAST_UPD_SRC,

CONVERT (VARCHAR (10),T1.DB_LAST_UPD, 101) + ' ' + CONVERT (VARCHAR (10),T1.DB_LAST_UPD, 8),

CONVERT (VARCHAR (10),T1.LAST_UPD, 101) + ' ' + CONVERT (VARCHAR (10),T1.LAST_UPD, 8),

CONVERT (VARCHAR (10),T1.CREATED, 101) + ' ' + CONVERT (VARCHAR (10),T1.CREATED, 8),
T1.LAST_UPD_BY,

T1.CREATED_BY,

T1.MODIFICATION_NUM,

T1.ROW_ID,

T1.BUILD_NUMBER,

T1.CURR_WEBFILE_VER,

T1.ENTERPRISE_NAME,

T1.PREV_WEBFILE_VER,

T1.RECORD_INFO_TEXT,

T1.REC_IDENTIFIER

FROM

 dbo.S_UIF_WEB_FILE T1

WHERE

 (T1.REC_IDENTIFIER = ?).

ObjMgrExtLangLog ObjMgrExtLangLog 0 000000b9475f1578:0 2007-12-12 07:46:29 [User: ] SQLBIND, 1, 1,
16:sdchs21n625:4330siebel.
```

Declaring Functions and Procedures in Siebel VB

The Siebel Compiler compiles Siebel VB functions and procedures in alphabetical order for each object definition. If a function or procedure calls another function or procedure that is not defined, then the compiler creates an error message that is similar to the following:

```
function_name Is An Unknown Function
```

To avoid this error, you must use the Declare statement to declare the function or procedure in the general declarations section. Siebel eScript does not require you to declare these functions. For more information, see *Siebel VB Language Reference* .

Using the Script Profiler

This topic describes how to use the Script Profiler. It includes the following information:

- *Overview of the Script Profiler*
- *Enabling the Script Profiler and Line Profiler*
- *Running the Script Profiler*

Overview of the Script Profiler

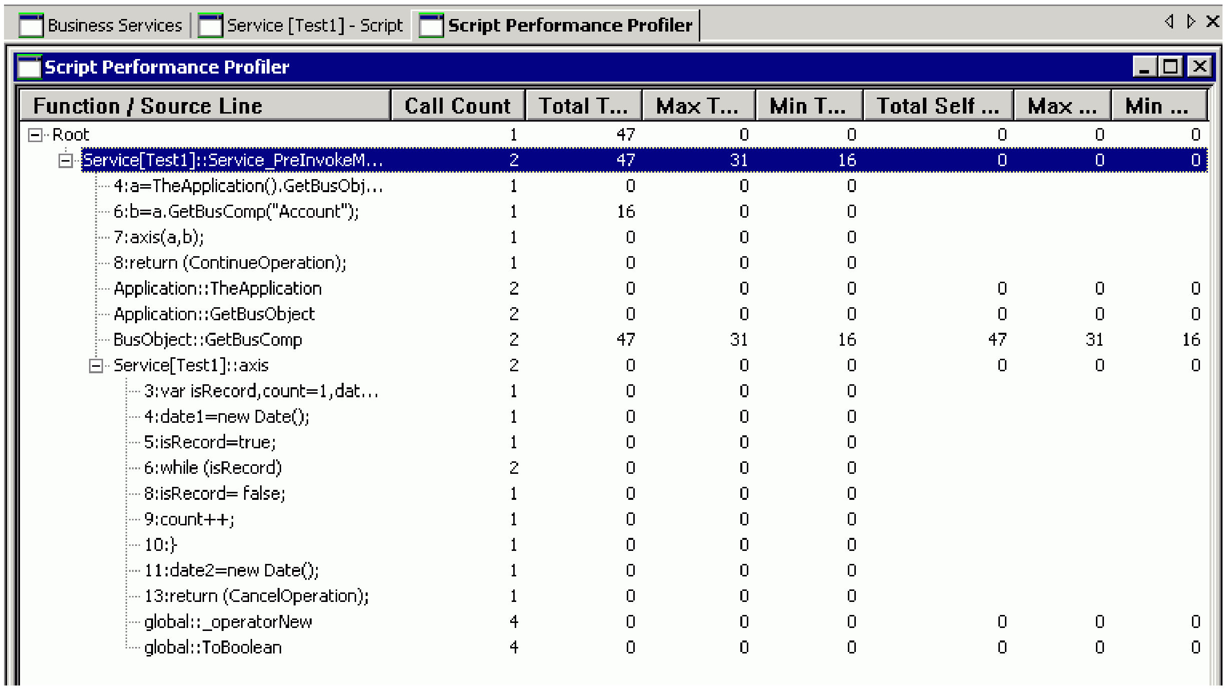
The *Siebel Script Performance Profiler* is a part of the eScript Engine that allows you to observe and monitor the performance of a script. For brevity, this guide refers to the Siebel Script Performance Profiler as the Script Profiler.

If you start Siebel CRM in debug mode, then the Script Profiler gathers and displays data for all executable scripts. It displays data in the Script Performance Profiler window and updates this data if a script runs. You can use this data to monitor script performance, identify performance bottlenecks, and compare performance with previous script runs. The Script Profiler allows you to do the following:

- **Use the Call Tree view.** Displays profile data as a tree of function calls. For more information, see *Example of the Script Performance Profiler Window*.
- **View function profile and line profile.** Line profile data includes the call count and total time spent for each line that runs in a function. Line profile information is available only for a compiled script and is not available for a line that does not run.
- **Save profile data to file.** You can save profile data that the Script Performance Profiler window shows to a text file.
- **Use the Siebel Script Debugger and the Siebel Script Profiler.** You can use the Script Profiler and the Script Debugger at the same time. Profile data is consistent even if the function uses Debugger functionality, such as a breakpoint.
- **View the script source.** The objects that a script references are opened in the Script Editor window. You can double-click a function name or line number to view the script from the Script Performance Profiler window. The View Source option is available only for a compiled script. It is not available for a runtime script.

Example of the Script Performance Profiler Window

The following image displays an example of the Script Performance Profiler window. It allows you to examine script performance. It displays a line number for each code line, total time spent to run that line, the number of times Siebel CRM runs the line, and so on.



Function / Source Line	Call Count	Total T...	Max T...	Min T...	Total Self ...	Max ...	Min ...
Root	1	47	0	0	0	0	0
Service[Test1]::Service_PreInvokeM...	2	47	31	16	0	0	0
4:a=TheApplication().GetBusObj...	1	0	0	0			
6:b=a.GetBusComp("Account");	1	16	0	0			
7:axis(a,b);	1	0	0	0			
8:return (ContinueOperation);	1	0	0	0			
Application::TheApplication	2	0	0	0	0	0	0
Application::GetBusObject	2	0	0	0	0	0	0
BusObject::GetBusComp	2	47	31	16	47	31	16
Service[Test1]::axis	2	0	0	0	0	0	0
3:var isRecord,count=1,dat...	1	0	0	0			
4:date1=new Date();	1	0	0	0			
5:isRecord=true;	1	0	0	0			
6:while (isRecord)	2	0	0	0			
8:isRecord= false;	1	0	0	0			
9:count++;	1	0	0	0			
10:}	1	0	0	0			
11:date2=new Date();	1	0	0	0			
13:return (CancelOperation);	1	0	0	0			
global::_operatorNew	4	0	0	0	0	0	0
global::ToBoolean	4	0	0	0	0	0	0

Description of the Columns in the Script Performance Profiler Window

The following table describes the columns that the Performance Profiler window includes. For the Source Line, Call Count, and Total Time columns, if line profiling is enabled for a function, then the line profile displays as child nodes of a function in the Call Tree view and in the Flat Profile view.

Description of Columns in the Script Performance Profiler Window

Column	Description
Function/Source Line	Function name and the name of the object that contains the function. For example: `Service [ATP Check]::service_preinvokeMethod and BusComp[Account]::foo`
Call Count	The number of times Siebel CRM calls a function. You can use one of the following views: • **Flat Profile view.** Displays the total number of times Siebel CRM calls the function. • **Call Tree view.** Displays the number of times Siebel CRM calls the function in the current position in the call tree. It displays profile data as a tree of function calls. A node represents each function in a call sequence. You can drill down into each node to examine a subtree. You can use the Expand All option in the Profiler toolbar to expand all nodes.
Total Time	Total milliseconds spent in this function and in nested functions.
Max Time	Maximum milliseconds spent in this function and in nested functions.
Min Time	Minimum milliseconds spent in this function and in nested functions.
Total Self Time	The total milliseconds spent in the current function, not including time spent in the subtree of this function.
Max Self Time	The maximum milliseconds spent in the current function, not including time spent in the subtree of this function.
Min Self Time	The minimum milliseconds spent in the current function, not including time spent in the subtree of this function.

Enabling the Script Profiler and Line Profiler

You must enable the Script Profiler and the line profiler; they are not enabled by default.

To enable the Script Profiler

1. Make sure your environment is currently connected to the database that a Siebel application uses, and that this application is opened in debug mode.

   You can use the Script Profiler only if your environment is currently connected to the database that the Siebel application uses, and if this application is opened in debug mode.
2. Set the debug options on the Debug tab of the Development Tools Options dialog box.

   Make sure you complete options in the Run-time Start Up Information section and the Login Information section. For more information, see *Setting Debug Options for Siebel Server Script Debugger*.
3. Make sure the Enable Profiler check box on the Debug tab of the Development Tools Options dialog box contains a check mark.

4. Enable line profiling:

    a. Click Set Line Profile Rules.

       The Set Line Profile Rules button resides in the Profiler Start Up Options section on the Debug tab of the Development Tools Options dialog box.

    b. In the Line Profile Rules dialog box, add a rule using information from the following table, and then Click Add.

Item	Description
Object Type	Choose the object type. Only scriptable objects are available. For more information, see *Object Types You Can Script*.
Object Name	Enter the name of an object. You can include the entire object name or use wildcard operators.
Function	Enter the name of a function. You can enter the entire function name or use wildcard operators. For example, you can enter `Service_PreInvokeMethod` or `Service*`.

    c. Repeat Step b for each rule you want to add.
    d. Click OK to exit the Line Profile Rules dialog box.

5. Click OK to exit the Development Tools Options dialog box.

   The settings for the line profile rule remain active only for the current session.

   To disable line profiling, make sure the Enable Profiler check box on the Debug tab of the Development Tools Options dialog box does not contain a check mark.

Running the Script Profiler

This topic describes how to run the Script Profiler. For more information, see *Overview of the Script Profiler*.

To run the Script Profiler

1. Make sure the Script Profiler is enabled.

   For more information, see *Enabling the Script Profiler and Line Profiler*.

2. Open the Siebel Script Editor for a scriptable object, and then deliver the changes to the script to a repository.

   Make sure you deliver the scripted objects or the projects that these objects reference to the repository that the Siebel client uses. If you do not deliver these objects, then the Script Profiler does not display them. For more information, see *Object Types You Can Script* and *Delivering Workspaces*.

   The Script Profiler is now enabled for runtime sessions and you can open the Siebel application in debug mode.

3. Click the Debug menu, and then click Start.

   This step opens the Siebel application in debug mode. If Siebel CRM prompts you for a login, then enter the user name and password, and then click Run. For example, to run a script for the Business Service Simulator, you query for the Service Name and Method Name, and then click Run.
4. Click the View menu, click Profiler, and then click Call Tree.

   The functions that Siebel CRM calls when it runs a script as a hierarchy are displayed in the Script Performance Profiler window. It displays the functions that it calls from other functions as child objects. For an example, see *Example of the Script Performance Profiler Window*.
5. In the Script Performance Profiler window, you can do the following:
   - Navigate between nodes to view the different parameters for a function.
   - Right-click a node, and then choose a menu item. For more information, see *Navigating in the Script Performance Profiler Window*.
6. (Optional) To set or reset line profile rules for a function, do one of the following:
   - Right-click a function node, and then click Enable Line Profiling or Disable Line Profiling. If you modify a line profile rule, then this modification applies to future calls to the function that this rule references. For more information, see *Enabling the Script Profiler and Line Profiler*.
   - Use the Debug tab in the Development Tools Options dialog box. For more information, see *Setting Debug Options for Siebel Server Script Debugger*.
7. Continue monitoring your script.

   For example, you want to test custom script on a business service. In the Siebel client, you can navigate to the Business Service Simulator to run a business service, and then navigate back to the Profiler window to examine the profile data that Siebel CRM logs while the script runs.

Navigating in the Script Performance Profiler Window

You can right-click in the Script Performance Profiler window, and then click one of the following menu items. The menu items that are enabled depends on the script. Some of these items might not be available for all scripts:

- **Expand Node**. Expands the function node.
- **Expand All**. Expands the entire tree of a function node.
- **Export**. Exports profile data to a text file that you specify.

  You can use Expand All and Export in the Profiler toolbar. For more information about the Profiler toolbar, see *Example of the Script Performance Profiler Window*.
- **View Source**. Navigates to the current function in the script. If a Script Editor window is not open for this function, then a new editor window opens and the cursor is placed at the beginning of this function.
- **Enable Line Profiler.** Enables a function for line profiling. If a function is not in the list of functions chosen for line profiling, then you can click Enable Line Profiler to enable it.
- **Disable Line Profiler.** Disables a function for line profiling.

Using the Script Profiler in Web Tools

The *Siebel Script Performance Profiler* is used with the eScript ST engine to observe and monitor the performance of a script. It is designed to be used in conjunction with the Siebel Script Debugger and gathers data only during a Siebel

Script debugging session. While very basic information will be gathered if you have not set up line debugging rules, if a script executes, there will be an entry in the Script Profiler Window. You must set up Line Profiling Rules to get detailed information about the lines that have executed. The columns in the Script Profiler Call Tree Pane are the same that exist in Siebel Tools. To see more go to *Example of the Script Performance Profiler Window*.

Note: The Script Profiler does not profile Browser Script. It only works on server eScript.

Setting up Line Profiling Rules

To gather detailed information about individual methods and their constituent lines, you must set up Line Profiling Rules. Once you have enabled the Script Profiler by setting the *Enable Profiler* check box to true (checked) you can set Line Profile Rules. It is best to set up Line Profiling Rules before your debug session starts, but if you add rules after it starts, they will be picked up the next time the scripts are executed.

1. Click **Set Profile Rules**.
2. Choose the object type for the scripted object. The choices are:
   a. **Application** (Server Script for an Application)
   b. **Service** (Server Script for a Business Service)
   c. **BusComp** (Server script for a Business Component)
   d. **WebApplet** (Server Script for an Applet)
3. Type in the name of the object in the `Object Name` field. You can use wildcards in the name to include more than one object of the same type.
4. Type the method name in the `Function` field. You can use wildcards to include more than one method from the objects.
5. Click **Add** to add your rule to the list of Line Profile Rules.

   Note: You can also delete rules from this dialog by selecting a rule and clicking Delete.
6. Do these steps for all the objects and methods you wish to profile.
7. When finished click OK.
8. Click OK to close the **Debugger Settings** dialog.

Note: If you misspell the object type or method you will not get any results for that object. If you choose the wrong Object Type to associate with a method, you will get only general results for that object, no individual lines will be profiled.

Using the Script Profiler in Web Tools

If you have enabled the Script Profiler prior to starting a script debugging session, using the Script Profiler is as simple as starting a debugging session and ensuring the scripts you wish to profile are executed during that session. The Script Profiler pane will show under the script debugger panes.

Note: You can undock the script debugger panes to view them in a larger screen. The Script Profiler pane will remain at the bottom of the panes whether docked or undocked. You may have to scroll to see all the contents of the Script Profiler pane.

1. Enable the Script Profiler.
2. Set up any Line Profile Rules you need.
3. Start a debugging session.

4. Trigger the scripts that you wish to execute.
5. Click the Call Tree icon in the script debugger toolbar to open the Script Profiler Call Tree Pane. The button is a toggle button and will also close the Script Profiler Call Tree Pane.
6. In the Script Profiler Call Tree Pane, you will see the results of all the scripted objects that have executed. If you configured Line Profile Rules, you can expand or collapse each object and its child methods.
7. If you have enabled Line Profile Rules for a method the call tree will contain the lines of each method that were executed.

   Note: If you did not set up any Line Profile Rules for a particular object and have started your debugging/profiling session, you can click the check box next to the arrow that expands/collapses the call tree for that object. That particular profile rule will be automatically added to the list of Line Profile Rules in Settings. The next time that object executes a script, it will show the lines of script individually. This is a shortcut to creating Line Profile Rules ahead of time.

   Note: If a line (such as a comment or some logic in a switch, if, or other block) is not executed in a script it will not display in the Call Tree. Only lines that are executed show in the Call Tree.

8. Click on an individual line in a method and it will open that script in the middle script debugging pane with that line highlighted. If you open more than one script, you can switch between them in this middle pane where they are displayed.
9. Click the arrow next to an object or method in the call tree when it points horizontally to expand that object. Click on the arrow when it points downward collapses that object and its tree.
10. If the debugging session continues, data for scripted objects will collect in the Script Profiler pane.
11. When your debugging session ends, the script profiling also stops.
12. Print the profiling data by clicking the printer icon at the very top of the Call Tree in the **Root** node. This allows you to print the contents of the Script Profiler Pane.

8 Customizing Objects

Customizing Objects

This chapter describes how to customize objects. It includes the following topics:

- *Overview of Customizing Objects*
- *Creating and Modifying Objects*
- *Validating Objects*
- *Examining Objects*

Overview of Customizing Objects

You can edit Workspace-enabled repository objects directly without locking any project. Note that only Workspace-enabled repository objects can be edited inside Workspaces. By default, all Workspace-enabled objects are locked at the database level. Therefore, you are not required to lock projects when you edit the objects or perform any type of CRUD (create, read, update, delete) operation.

Note: Locking a project is still required before you edit the non-Workspace objects. The non-Workspace objects are Table, Repository, Type, EIM Table, Projects, Dock objects, Schema Maintenance, and Server Components. For more information, see *Using Workspaces*.

Creating and Modifying Objects

This topic describes how to create, modify, rename, copy, or delete an object. It includes the following information:

- *Creating Objects*
- *Modifying Objects*
- *Renaming Objects*
- *Copying Objects*
- *Deleting Objects*

This task is a step in *Roadmap for Setting Up and Using Siebel Tools*.

Creating Objects

You can use a wizard to create an object, or you can create it manually. It is recommended that you use a wizard. For example, to create a new business component, you use the Business Component Wizard. A wizard guides you while you configure a new object. It prompts you for the required property values and automatically configures any required child objects. If a wizard is not available for the object type that you want to create, then you can create it manually in the Object List Editor. For more information about using wizards and creating objects, see *Configuring Siebel Business Applications* .

Using a Wizard to Create Objects

This topic describes how to use a wizard to create an object.

To use a wizard to create objects

1. Click New Object Wizards (the magic wand icon) on the application banner in Web Tools.

   Note: In Siebel Tools, click File, click New Object, and then select one of the following as required in the dialog that opens: General, Applets, EAI, Task. If you select EAI, the following EAI wizard options are available: Integration Object, OLEDB Rowset, Web Service, Data Access Service.

2. Select the wizard that you want to create the new object and follow the instructions that the wizard shows.

   For more information, see *Using the New Object Wizard* and *Using IDE in Web Tools*

Using the Object List Editor to Create New Objects

This topic describes how to use the Object List Editor to create a new object. For more information, see *Locating Object Definitions in the Object List Editor*.

To use the Object List Editor to create new objects

1. In the Object Explorer, click the object type that you want to create.
2. Click New (the plus icon) on the applet banner in Web Tools.

   Note: In Siebel Tools, right-click the Object List Editor and then click New Record.

3. Enter property values in the new row in the Object List Editor.

   You must enter values in the Name and Project properties. You might also be required to set other properties, depending on the object type.
4. To save your modifications, click anywhere outside of the new row.

Modifying Objects

You can use the Object Explorer and Object List Editor (see *About the Object Explorer and Object List Editor*) to locate an object and then modify its object definitions (or properties).

For an example showing how to locate the object definitions of the Siebel Field Service application object type and then modify one of its properties, see *Locating Object Definitions in the Object List Editor*.

For guidelines about when to modify an object or when to create a new object, see *Configuring Siebel Business Applications* .

Renaming Objects

It is recommended that you copy an object, and then rename the copy instead of only renaming the original object. If you rename an object, then a message similar to the following is returned:

```
Modifying the name of a checked out or locked object causes "unique constraint"
error during check-in. To avoid this error, modify the name of the object back to the
original name. Do you want to continue?
```

For more information, see *Copying Objects*.

Copying Objects

Note: This task applies only to Siebel Tools.

This topic describes how to copy an object. For guidelines about copying an object, see *Configuring Siebel Business Applications* .

To copy an object

1. Create or open a Workspace.
2. In the Object List Editor, locate the object that you want to copy.

   For more information, see *Locating Object Definitions in the Object List Editor*.
3. Click the Edit menu and then click Copy Record.

   Siebel inserts a new record preceding the record that you copied. This new record includes all of the same values as the record that you copied, except the Name property is empty and the Changed property contains a check mark. Siebel also creates a copy of any child objects for the record you copied. For example, if you copy the Account business component, then it creates a new business component that includes several hundred business component fields.
4. Enter a new value for the Name property.
5. If necessary, modify other properties and child objects.
6. To save your modifications, click anywhere outside of the new row.

Deleting Objects

Note: This task applies only to Siebel Tools.

This topic describes how to delete an object.

To delete objects

1. Create or open a Workspace.
2. In case of non-Workspace enabled objects, in the Object List Editor, locate the object that you want to delete.

For more information, see *Locating Object Definitions in the Object List Editor*.

> **Note:** In case of Workspace-enabled objects, you cannot delete any predefined object. If you try to delete the object, it will return an error indicating that deletion is possible only for objects created in the current version and that you must inactivate the record to proceed further. For more information on making an object inactive, see *About the Inactive Property*.

3. Ensure that either the project or the object is locked.
4. Click the Edit menu and then click Delete Record.

> **CAUTION:** It is recommended that you do not delete a predefined object. Other objects might reference this predefined object. Instead, you can make this object inactive. For more information, see *About the Inactive Property*.

The object and all its children are deleted. For example, if you create a business component named My Account Business Component and you delete it, then Siebel deletes all child objects (such as fields) of My Account Business Component . It does not delete objects that only reference the object that you delete. For example, if you delete a view, then it does not delete the applets that reference this view.

Validating Objects

This task is a step in *Roadmap for Setting Up and Using Siebel Tools*.

The *Validate Tool* is an error correction tool that you can use to ensure each object that you customize has no configuration errors. For example, you can use the Validate Tool to make sure an error Workflow Process does not itself contain an error Workflow Process. When you validate a Workflow Process, warnings will be returned describing any errors, if there are any. The Validate dialog box lets you correct the errors. You can keep the Validate dialog box open to view the error messages (since the dialog is not modal) while you correct the problems, or you can proceed to deliver the Workflow Process without fixing the validation errors.

Validation uses a set of rules that help make sure your configuration changes are logically consistent with other objects. If you validate object_A, then the child objects of object_A are also validated. For example, if you validate the Account business component, then the child objects of the Account business component (such as its fields, joins, and so on) are also validated.

> **Note:** This is not true for Workspaces. Although a Workspace can have child Workspaces, the Validate Tool examines only the specific Workspace that you select in the Workspace Explorer (and ignores any child Workspaces).

Using the Validate Tool reduces the time needed to deliver Workspaces and test application functionality and ensures best practices for basic configuration. You can use the Validate Tool to validate the following:

1. A single object or multiple objects within an object type – see *Single Object Type Validation*.
2. Workspaces – see *Workspace Validation*.
3. An entire repository – see *Batch Validation*.

Single Object Type Validation

You can use the Validate Tool to validate a single object or multiple objects within an object type. The following procedure shows how to validate objects using the Object List Editor.

To validate objects using the Object List Editor

1. In the Object Explorer, locate and select one or more objects of the same type to validate.

   You can only select objects from a single object type to validate. For example, select multiple Business Components or Applets or Applet Controls. You cannot select a Business Component AND an Applet.

   You can select child objects to validate (for example, the Fields of a Business Component or the Methods of a Business Service), and if so, only the Fields and their child objects will be validated. The entire Business Component will not be validated.

   For more information, see *Locating Object Definitions in the Object List Editor*.
2. Click the Applet Menu (the cogwheel icon) and then select the Validate option.

   The Validate dialog box opens – see *Elements of the Validate Dialog Box*.
3. In the Validate dialog box, click Options.

   The Validation Options dialog box opens where you can review the validation rules that are available and specify which rules to enforce or ignore during validation. For more information, see *Elements of the Validation Options Dialog Box*.
4. In the Validation Options dialog box:

   a. Select/deselect the check box next to each rule that you want to enforce/ignore.
   b. (Optional) Click Advanced Options to open the Advanced Options dialog where you can select objects to exclude from (object type, Workspace or batch) validation.
   c. (Optional) Once the list of rules to enforce and ignore is set, click Save to save the information to a text file. This helps if you need to review or share the information later with your team.
   d. (Optional) Select the check box next to the following options as required:

      - **Do Not Report Warnings.** Select this option to report only errors and ignore warnings.
      - **Abort Validation After <x_number> Errors.** Select this option and then enter a number (for example 5) so that object validation will stop after <x_number> of errors are identified.

   e. Click OK.

      Note that if you select Ignore All or deselect all the rules in the Validation Options dialog, then you have chosen to ignore the rules for this object and will receive the following message:

      ```
      There are no rules currently enforced for the selected object(s).
      You may update the list of rules to be enforced from the 'Options' button.
      ```
5. In the Validate dialog, click Start to start the validation process.

   Siebel starts validating the object and returns the results in the Validate dialog when finished:

   - For example:

     `Total tests failed: 0` or

     `Total tests failed: 3`
   - Validation Errors and Warnings appear in the Validate list. Each row in the Validate list identifies a rule violation (Warning or Error) for the object.
   - Select an error in the Validate list and then click Go To to drill down on the object that causes the error.
   - (Optional) Click Save As to save the validation results to a `validation.log` file on your local machine.

Depending on your browser naming conventions, the validation results will be saved to `validation(1).log` if *validation.log* already exists or `validation(2).log` if *validation(1).log* already exists, and so on.

Workspace Validation

You can use the Validate Tool to validate Workspaces including the following:

- All objects changed in the current Workspace.
- All objects changed in a particular version of the current Workspace.
- All objects changed in all versions of the current Workspace.

Only the selected (or current) Workspace will be validated – child Workspaces will not be validated. The following procedure shows how to use the Validate option on the Workspace Toolbar to validate a Workspace.

To validate a Workspace

1. Go to the *Workspace Dashboard* and open the Workspace that you want to validate.
2. Click Validate on the Workspace Toolbar. Note the following:
    - The Validate option on the Workspace Toolbar will be disabled if the Workspace that you want to validate is not currently open.
    - If you select a row in the Workspace Explorer that does not match the currently open Workspace, then the Validate option on the Workspace Toolbar will also be disabled. Hence make sure to select the row in the Workspace Explorer that matches the currently open Workspace.

   The Validate dialog box opens – see *Elements of the Validate Dialog Box*.
3. In the Validate dialog box, click Options.

   The Validation Options dialog box opens where you can review the validation rules that are available and specify which rules to enforce or ignore during validation. For more information, see *Elements of the Validation Options Dialog Box*.
4. In the Validation Options dialog box:
    a. Select the check box next to each rule that you want to enforce and deselect the check box next to each rule that you want to ignore.
    b. (Optional) Click Advanced Options to open the Advanced Options dialog where you can select objects to exclude from (object type, Workspace or batch) validation.
    c. (Optional) Once the list of rules to enforce and ignore is set, click Save to save the information to a text file. This helps if you need to review or share the information later with your team.
    d. (Optional) Select the check box next to the following options as required:
        - **Do Not Report Warnings.** Select this option to report only errors and ignore warnings.
        - **Abort Validation After <x_number> Errors.** Select this option and then enter a number (for example 5) so that object validation will stop after <x_number> of errors are identified.
    e. Click OK.

       Note that if you select Ignore All or deselect all the rules in the Validation Options dialog, then you have chosen to ignore the rules for this Workspace and will receive the following message:

```
There are no rules currently enforced. You may update the list
```

```
of rules to be enforced from the 'Options' button.
```

5. In the Validate dialog, click Start to start the validation process.

   Siebel starts validating the object and returns the results in the Validate dialog when finished:

   - For example:

     `Total tests failed: 0` or

     `Total tests failed: 3`
   - Validation Errors and Warnings appear in the Validate list. Each row in the Validate list identifies a rule violation (Warning or Error) for the object.
   - Select an error in the Validate list and then click Go To to drill down on the object that causes the error.
   - (Optional) Click Save As to save the validation results to a `validation.log` file on your local machine.

     Depending on your browser naming conventions, the validation results will be saved to `validation(1).log` if *validation.log* already exists or `validation(2).log` if *validation(1).log* already exists, and so on.

Troubleshooting Workspace Validation

Because there can be more than one version of a Workspace, you can validate either the most current version of the Workspace or all versions of the Workspace. As a result, note the following when validating Workspaces:

- You will be prompted to answer the following message before being allowed to continue with the validation process if a Workspace has previous versions:

```
<-- Workspace Validation Options -->

You've selected Workspace dev_sadmin_rep_validation version 4.
Please select one of the following validation options:
- Objects changed in selected version only.
- Objects changed in this and all earlier versions of this Workspace.

 OK / Cancel
```

- If a Workspace contains no changed objects, then there is nothing to validate so you will be prompted to select a different Workspace to validate. For example:

```
No changes have yet been made in this Workspace.
```

Batch Validation

You can validate an entire repository using batch validation. Batch validation can be a very time-consuming and lengthy process, especially if you opt to validate all objects in a repository. One way to restrict the volume of objects to validate is in the Web Tools client – by selecting objects to exclude from the validation process in the Advanced Options dialog and saving the preferences. When batch validation starts, it reads these preferences and excludes the objects selected in the Web Tools client from validation – that is, provided the `ComponentName` parameter is set in the configuration file.

The following procedure shows how to use the command line to batch validate all objects in a repository. Batch validation is only available through the command line. While batch validation is running, you can continue to work in Web Tools.

To validate all objects in a repository

1. Set the following in the [Siebel] section of the configuration (.cfg) file to use the exclusions defined in the Advanced Options dialog in the Web Tools client:

   ```
   [Siebel]
   ComponentName = Siebel Web Tools Client
   ```

2. Open a command line, and then navigate to the `$SIEBEL_HOME\BIN` folder.
3. Use the `siebdevcli.exe /BatchValidate` or `siebdevcli.exe /BV` command option (they are equivalent) to run batch validation. The syntax to use is as follows:

   ```
   siebdevcli.exe  /c config_file /d data_source /u user_name /p password /BatchValidate
   /exportfile file_name
   ```

   You can use either `/exportfile` or `/f` (they are equivalent) to specify the file name. This switch is also optional. For example:

   ```
   siebdevcli.exe  /c swtools.cfg /d ServerDataSrc /u sadmin /p ****** /BatchValidate
   /exportfile c:\validation.log
   ```

   or

   ```
   siebdevcli.exe  /c swtools.cfg /d ServerDataSrc /u sadmin /p ****** /BatchValidate
   /f c:\validation.log
   ```

   The `/BatchValidate` switch runs all validation rules for the entire repository.

Elements of the Validate Dialog Box

The following table describes the elements of the Validate dialog box.

Label/Field	Element	Description
Validate	List (Validate list)	Shows the results of the validation process. Each row in the Validate list identifies a rule violation for the object. • Click a column heading to sort the rows. • Select and move the border of a column heading cell to resize the columns.
Severity	Column	Indicates one of the following types of errors (and its corresponding icon): • **Warning.** The *warning icon* is a backslash (\) in a circle, which appears in the color Yellow. • **Error.** The *error icon* is an exclamation mark (!) in a red triangle. This error might cause a problem in Siebel CRM at runtime.
Rule	Column	Displays an integer that identifies the number of the violated rule. Rules are listed sequentially according to the rule number.
Object	Column	Displays the name of the object that failed validation.

Label/Field	Element	Description
Description	Column	Displays a description of the error or warning.
None	Box (validate error details)	Displays more information about the error or warning message selected in the Validate list.
Options	Button	Click to open the Validate Options dialog where you can specify the rules to enforce during validation. For more information, see *Elements of the Validation Options Dialog Box*.
Start	Button	Click to start the validation process.
Go To	Button	Select an error in the Validate list and then click Go To to drill down on the object that causes the error.
Cancel	Button	Click to cancel the validation process or to close the Validate dialog.
Load Existing Validation Log	Field	Click Browse to select a validation log (that you previously saved) and then click Load to load it back into the Validate Tool.
Load	Button	Click to load the contents of the log file that is specified in the Load Existing Validation Log field back into the Validate Tool.
Save As	Button	Click to save the list of errors, that currently appear in the Validate list, as a log file.

Elements of the Validation Options Dialog Box

The following table describes the elements of the Validation Options dialog box where you can review the rules that are available and specify which rules to enforce or ignore during the validation process.

Label/Field	Element	Description
Validation Options (Enforce Checked Rules For Validation)	List	Lists the rules that you can enforce during validation. Click a column heading to sort the rows. Select and move the border of a column heading cell to resize the columns.
Severity	Column	Indicates one of the following types of errors (and its corresponding icon): • **Warning.** The *warning icon* is a backslash (\) in a circle, which appears in the color Yellow. • **Error.** The *error icon* is an exclamation mark (!) in a red triangle. This error might cause a problem in Siebel CRM at runtime.

Label/Field	Element	Description
Rule	Column	Displays an integer that identifies the number of the violated rule. Rules are listed sequentially according to the rule number.
Object	Column	Displays the object type to examine.
Description	Column	Displays a description of the rule.
Enforce	Column (check box)	Indicates whether the rule is enforced or not as follows: • If this check box is selected, then the rule will be enforced. All objects, as specified by the Object column, will be validated. • If this check box is not selected, then the rule will not be enforced. If you click the Enforce All or Ignore All button in the Validation Options dialog, then the value in the Enforce column changes accordingly.
None	Box (validation rule details)	Displays more information about the validation rule selected in the Validation Options list.
Save	Button	Once the list of rules to enforce or ignore is set, click Save to save the information to a text file. This helps if you need to review or share the information later with your team. If you press ENTER, and if the Validation Options dialog is open, then other settings are saved to a preferences file.
Enforce All	Button	Click to enforce all rules in the Validation Options list. Doing this selects the check box in the Enforce column for all rules.
Ignore All	Button	Click to ignore all rules in the Validation Options list. Doing this deselects the check box in the Enforce column for all rules and Siebel will not validate any objects.
Advanced Options	Button	Click to open the Advanced Options dialog where you can select objects to exclude from validations in Web Tools and the batch validation process.
OK		Click to accept and apply the specified validation options to the validation process.
Cancel		Click to cancel validation options or to close the Validation Options dialog.
Do Not Report Warnings Option	Check box	Select this check box to report only errors (and not warnings). Doing this also deselects the check box in the Enforce column for all rules with a Warning severity.
Abort Validation After <x_number> Errors	Check box	Select this check box and then enter an <x_number> so that object validation will stop after <x_number> errors are identified.

Examining Objects

This topic describes how to examine objects. It includes the following information:

- *Searching the Repository*
- *Viewing Object Relationships*
- *Comparing and Synchronizing Objects Between Repositories and Archives*
- *Comparing Different Versions of a Workflow Process or Task UI*
- *Determining When Siebel Created or Updated a Record*

Searching the Repository

This topic describes how to locate object definitions for multiple object types. For information about how to locate object definitions for a single object type, see *Locating Object Definitions in the Object List Editor*.

If you know that one or more properties includes a value, then you can search the repository across all properties of multiple object types to locate the objects that include this value.

To search the repository (Siebel Tools)

1. Click the Tools menu and then click Search Repository.

   Searching the repository can consume a significant amount of time. You can click Cancel in the Search Repository dialog box to cancel a search at any time while the search runs.
2. In the Search Repository dialog box, in the Parameters section, enter the search criteria in the Search value window.
3. To limit the search to values that are case-sensitive, make sure the Case Sensitive check box contains a check mark.

   For example, select this check box to locate objects that include a value of Account and not account.
4. To limit the search to values that match the entire search string that you enter in Step 2, make sure the Exact Match check box contains a check mark.
5. In the Types to Search list, choose the object type that Siebel must search for.

   Note the following:

   - All object types are searched, by default.
   - Click the object type to choose a single object type.
   - Use multiselect to choose multiple object types.

     For more information, see *Selecting Multiple Records in the Object List Editor*.
   - For improved performance, search only the minimum number of object types.

6. Click Search Now.

   The returned search results appear in the window at the end of the Search Repository dialog box. The window lists all objects that match the search criteria. The window includes the following columns:

Column	Description
Type	Object type of the object that the search returns.
Name	Name of the object that the search returns.
Property	Property name of the object that the search returns.
Value	Property value of the object that the search returns.

7. To display the object definition of an item that the search returns, double-click this item in the results window.

   The object definition of the item is displayed in the Object List Editor.
8. To export the search results to a file, click Export.

Viewing Object Relationships

You can use a visualization view to examine how objects relate to each other.

To view object relationships (Siebel Tools)

1. Set options for the visualization views as follows:
   a. Click the View menu and then click Options.
   b. In the Development Tools Options dialog box, click the Visualization tab, and then set the options.

      For more information, see *Development Options for Visualization Views*.

2. To open a visualization view, do one of the following:

   - Click the View menu, click Visualize, and then click one of the following menu items:
     - View Details
     - View Relationships
     - View Descendents
     - View Web Hierarchy
   - Right-click an object in the Object List Editor and then select a Visualization view.

   The following table describes the Visualization views.

View	Description
Details	Displays a diagram that illustrates relationships depending on the following object type that you choose in the Object List Editor: ◦ **Business Component.** Illustrates the relationship between this business component and tables that this business component references. ◦ **Business Object.** Illustrates the relationship between this business object and the links and business components that this business object references.
Relationships	Displays a diagram that illustrates relationships depending on the following object type that you choose in the Object List Editor: ◦ **Business Component.** Illustrates the relationship between this business component and links to other business components. ◦ **Table**. Illustrates the relationship between this table and other tables.
Descendents	Displays a dialog box that lists all objects that reference the current object in their Upgrade Ancestor property. For example, if you right-click the Proposal Template Section business component and click View Descendents, then Siebel lists the Proposal Template Section Hierarchy business component. This is the only business component that includes a Proposal Template Section in the Upgrade Ancestor property.
Web Hierarchy	Displays a diagram that illustrates a Web hierarchy depending on the following object type that you choose in the Object List Editor: ◦ Applet ◦ Application ◦ Business Component ◦ Screen ◦ View The Web hierarchy displays the parent-child relationships between the object you choose and the parent and child objects of this object throughout the hierarchy.

Comparing and Synchronizing Objects Between Repositories and Archives

You can compare two objects that are of the same object type. Siebel Tools uses color-coded icons to highlight differences. You can choose and copy properties and individual child objects from one object to another object. You can use this feature to propagate a modification that you make to an ancestor object to the descendents of this object or to other objects that are of a similar type. You can assess and adjust differences between objects. You can also compare properties of checked out objects with their counterparts on the Siebel Server. For more information about ancestor objects, see *Configuring Siebel Business Applications* .

You can compare two objects that are of the same type. The Object Comparison dialog box displays a line-by-line comparison between the two objects. You can compare the top-level objects that are defined in the following items:

- Current repository
- Different repositories
- Archive files

Note: You can compare and synchronize objects between repositories and archives when Workspaces are not enabled (a flattened repository) and the concept of Workspace is not applicable.

For information about top-level object types, see *Displaying Object Types in the Object Explorer*.

Comparing Two Objects in the Same Repository

You can compare two objects that reside in the same repository.

To compare two objects in the same repository (Siebel Tools)

1. In the Object Explorer, select an object type.
2. In the Object List Editor, select two top-level object types.

   For more information, see *Selecting Multiple Records in the Object List Editor*.
3. Click the Tools menu, click Compare Objects, and then click Selected.
4. Examine the results in the Compare Objects dialog box.

   For more information, see *Elements of the Compare Objects Dialog Box*.

Comparing Two Objects in Different Repositories

You can compare an object in the current repository to an object in a different repository.

Note: The objects that you compare must have the same name in both repositories. A workflow name is appended with the version number of the workflow. Consequently, to compare two workflows in different repositories, the workflows must have the same name and the same version number.

To compare two objects in different repositories (Siebel Tools)

1. In the Object Explorer, select an object type.
2. In the Object List Editor, select one top-level object type.

3. Click the Tools menu, click Compare Objects, and then click Selected vs. Repository.
4. In the Open Repository dialog box, select the repository that includes the object you want to compare, and then click Open.
5. Examine the results in the Compare Objects dialog box.

   This dialog box displays the following:

   - The object in the current working repository in the one window
   - The corresponding object in the other repository in the other window

     You can update the current working repository or the other repository from the Object Comparison dialog box. For objects that are not Workspace-enabled, to do an update, you must make sure the project that the object references is locked. For more information, see *Elements of the Compare Objects Dialog Box*.

Comparing Object in Current Repository to Object in Archive

You can compare an object in the current repository to an object in an archive.

To compare object in current repository to object in archive (Siebel Tools)

1. In the Object Explorer, select an object type.
2. In the Object List Editor, select one top-level object type.
3. Click the Tools menu, click Compare Objects, and then click Selected vs. Archive.
4. In the Select Archive File to Compare Against dialog box, select the archive file that includes the object you want to compare.

   An archive file is saved as a SIF file. For more information, see *Overview of Archiving Objects*.
5. Click Open.

   The comparison starts at the project level. It does the following:

   - If it finds a corresponding object in the archive, then it displays the Compare Objects dialog box.
   - If it does not find a corresponding object in the archive, then it does nothing.

   For more information, see *Elements of the Compare Objects Dialog Box*.

Comparing Objects in Two Different Archives

You can compare an object in an archive to an object in another archive.

To compare objects in two different archives (Siebel Tools)

1. In the Object Explorer, click an object type.
2. In the Object List Editor, choose one top-level object type.
3. Click the Tools menu, Compare Objects, and then click Archive vs. Archive.
4. In the Select Archive File for Left Side of Comparison dialog box, choose an archive file, and then click Open.
5. In the Select Archive File for Right Side of Comparison dialog box, choose an archive file, and then click Open.

   The Object Comparison dialog box opens and is populated with the contents of the archives you selected in Step 4 and Step 5. During the comparison, these archives are read-only. For more information, see *Elements of the Compare Objects Dialog Box*.

Synchronizing Objects Between Repositories

You can use the Compare Objects dialog box to synchronize objects.

To synchronize objects between repositories (Siebel Tools)

1. Lock the projects that the objects that you want to synchronize reference.
2. In the Object List Editor, choose two top-level object types of the same object type.

   For more information, see *Selecting Multiple Records in the Object List Editor*.
3. Click Tools, click Compare Objects, and then click Selected.
4. In the Compare Objects dialog box, choose an object in the First Selection box.

   For more information, see *Elements of the Compare Objects Dialog Box*.
5. Click the (right, forward facing) arrow button.

   Depending on whether or not a corresponding object exists in the repository that the Second Selection section represents, one of the following happens:

   - **Corresponding object exists.**The properties of the object in the repository that the Second Selection section represents are modified. The properties are modified so that they are equal to the object properties of the object that the First Selection section shows.
   - **Corresponding object does not exist.** A corresponding object is created in the repository that the Second Selection section represents.

   If you synchronize an object from one repository to another repository, then not only is the object synchronized but all child objects included in the object are synchronized.

Comparing Workflow Processes or Task UIs

Now that Workflow Process and Task UI are Workspace-enabled, legacy version numbers are no longer relevant. Legacy (Workflow Process or Task UI) records with version numbers are considered as individual, unique records and their names (Workflow Process Name or Task UI Name) must be unique.

Comparing legacy versions of a Workflow Process or Task UI is similar to comparing two separate Workflow Process or Task UI records. The comparison is also similar to *Comparing Workspace Versions*.

Determining When Siebel CRM Created or Updated a Record

You can determine the last time that Siebel created or updated a record and who made the modification or update.

To determine when Siebel created or updated a record

1. Create or open a Workspace.
2. In the Object List Editor, select the object that you want.

   For more information, see *Locating Object Definitions in the Object List Editor*.
3. Click the Help menu and then click About Record.

4. In the About Record dialog that appears, review the following information: Created On, Created By, Updated On, Updated By, Last Updated On, and so on.

9 Managing Repositories

Managing Repositories

This chapter describes how to manage repositories. It includes the following topics:

- *Overview of Managing Repositories*
- *Exporting and Importing Repositories*
- *Upgrading Repositories*
- *Process of Migrating Repositories*
- *Deleting Repositories*

Overview of Managing Repositories

A *Siebel Repository* is a set of tables that includes Siebel objects and server scripts. These objects and scripts define a Siebel application, such as Siebel Field Service. This repository provides the metadata that Siebel CRM requires to interact with enterprise data and to interact with the people who use the Siebel application. The Siebel application stores these tables in the Siebel runtime repository, and then reads the repository at run time. You view and modify the information that this repository contains – for more information, see *Configuring Siebel Business Applications* .

Siebel CRM includes *Incremental Repository Merge*, which is a feature that allows you to merge multiple repositories to a single repository during an incremental upgrade. It automatically does some of this upgrade work for you, such as importing SIF files and seed data, applying schema modifications, and deliver. For more information, see *Siebel Database Upgrade Guide* .

This topic includes the following information:

- *Files That You Use to Manage Repositories*
- *Viewing Information About the Current Repository*
- *Renaming Repositories*
- *Configuring Siebel CRM to Read Data from a Single Repository*
- *Restricting the Objects That Developers Can Modify*

Files That You Use to Manage Repositories

A Siebel Archive File (SIF) is a type of file that you can use to import object definitions into or export objects from the Siebel Runtime Repository. You can use SIF files to move your modifications from a source environment to a destination environment. SIF files are written in XML format, use the following hierarchy, and can include property values and scripts:

1. Repository
2. Project
3. Object
4. Child Objects

For example, the following code is part of a SIF file that is created when objects are added to a hotfix. It includes a definition for the Account Assoc Applet:

```
<REPOSITORY NAME="Siebel Repository" ... >

 <PROJECT ... NAME="Account (SSE)" ... >

 <APPLET ASSOCIATE_APPLET="Account Assoc Applet" BUSINESS_COMPONENT="Account"
 CLASS="CSSFrameListBase" ... NAME="Account List Applet" ... >
 <APPLET_METHOD_MENU_ITEM... >
 </APPLET_METHOD_MENU_ITEM>...
 </APPLET>

 <BUSINESS_COMPONENT CACHE_DATA="N" CLASS="CSSBusComp" ... >
 </BUSINESS_COMPONENT>
 ...

 </PROJECT>

</REPOSITORY>
```

For more information about how to export objects to a SIF file and import objects from a SIF file, see *Archiving Objects*.

Viewing Information About the Current Repository

Only one repository is available for all users working on Workspaces in the Siebel environment. This repository is open by default.

In some situations, multiple repositories might reside on the Siebel Server (for example, if you upgrade to a new version of Siebel CRM).

To view information about the current repository (Siebel Tools)

1. Click the File menu and then click Open Repository.

   The Open Repository dialog that appears lists the repositories in the database that Siebel uses. The repository that is currently being used is highlighted.
2. Select About SRF from the Help menu.

   The following table describes the information in the About Repository File dialog that opens.

Field	Description
Internal Version	Version number that Oracle maintains. Siebel CRM modifies this value only if the internal format of the repository is modified (for example, during an upgrade).
User Version	Reserved for use by Oracle's Siebel Anywhere. It maintains this number when Oracle creates kits that upgrade the repository. Siebel CRM reads this value when it does a version check.
Full Compile	Choose this option to display information about the most recent full compile.

Field	Description
Last Incremental Compile	Choose this option to display information about the most recent incremental compile. If no incremental compile has occurred since the last full compile, then this option is not available.
When	Date of the last compile.
Machine Name	Name of the computer that Siebel CRM uses to compile the repository.
Language	Language code.
User Name	The Microsoft Windows logon name of the user who compiled the repository.
Repository	Repository name of the repository that was current when Siebel CRM performed the compile. This value is typically Siebel Repository.
Tools Version	The version number and build number of the Siebel software that performed the compile. Global Customer Support can use this information to help you resolve a problem.
Schema Version	Database schema version of the Siebel database that Siebel CRM uses to compile the repository.

To view information about the current repository (Web Tools)

1. Select About Repository from the Help menu.

   The following information appears in the About Repository dialog that opens.

   - **Repository.** For example, Siebel Repository.
   - **Schema Version.** For example: 52.62.62.2
   - **Language.** For example: ENU.
   - **Runtime Version.** For example: Safe Boot
   - **Published Type.** For example: Full.
2. Click OK to close the dialog.

Renaming Repositories

It is recommended that you use Siebel Repository as the name of the repository that you use in your production environment. If you want to rename the repository, then it is recommended that you use the following procedure.

To rename a repository

1. Display the Repository object type.

   For more information, see *Displaying Object Types in the Object Explorer*.
2. In the Object Explorer, click Repository.
3. In the Repositories list, locate the repository that you want to rename.

   For more information, see *Locating Object Definitions in the Object List Editor*.
4. Enter the new name in the Name property, and then step off the record to save your modifications.

   For more information, see the following topic.
5. Modify the value of the Siebel Repository enterprise parameter to the new name of the repository.

   For information about how to modify an enterprise parameter, see *Siebel System Administration Guide* .
6. Modify the Application Main Repository Name parameter in the Object Manager.

Guidelines for Naming Repositories

If you name a repository, then it is recommended that you adhere to the following guidelines:

- Use a naming convention for all repositories that your Siebel CRM implementation uses. Siebel Servers reference a repository by name. The procedure that you use to upgrade to a new version of Siebel CRM depends on the repository name. A consistent naming convention promotes successful configuration and testing and minimizes the work required to migrate a new repository or to do an upgrade.
- Use the default Siebel Repository name, where possible. You can modify this name only if it is absolutely necessary. The default configuration that Siebel CRM uses assumes that the repository name is Siebel Repository.
- Use the same repository name in your test environment that you use in your production environment.
- If your development environment uses multiple repositories, then use a unique and descriptive name for each repository. For example, you might use Siebel v8.2 Original as the name of the repository when you install Siebel CRM. You can use another descriptive name for the repository that you use during an upgrade and for a repository from a prior custom configuration.

Configuring Siebel CRM to Read Data from a Single Repository

The example in this topic describes how to configure Siebel CRM to read data from a single repository when the Siebel database contains multiple repositories.

To configure Siebel CRM to read data from a single repository

1. Log in to the Siebel client with administrative privileges, and complete the following steps:

   a. Navigate to the Administration - Order Management screen, and then the Message Types view.
   b. Navigate to the Payload view, and then create a new record.
   c. In the new record, access the drop-down list for the Response Field.

      If the Response Field drop-down list:

      - **Does not display duplicate values.** The Siebel database does not contain multiple repositories, and you can exit this task.
      - **Displays duplicate values.**The Siebel database contains multiple repositories, and you must configure Siebel CRM to read data only from a single repository. Continue to Step 2.
   d. Delete the new record that you created.
2. Configure Siebel CRM to read data only from a single repository:

   a. Log in to Siebel Tools or Web Tools.
   b. In the Object Explorer, click Pick List.
   c. In the Pick Lists list, query the Name property for the following value:

      ```
      UMS PickList Response Field
      ```
   d. Modify the Search Specification property using the value from the following table.

Property	Value
Search Specification	`[Buscomp] = 'UMS Response' and [Repository Id] = RepositoryId()`

      Siebel CRM displays the UMS Type Variables List Applet in Step 2. This applet references the UMS Type Variable business component. It displays the Response Field Name field and uses the UMS PickList Response Field picklist for this field. This picklist uses the following search specification when it references the UMS Pick List Field business component:

      ```
      [Buscomp] = 'UMS Response'
      ```

      The UMS Pick List Field business component references the S_FIELD table. It does not include a search specification, so Siebel CRM does not filter the records that it gets according to the specifications that the repository contains. If the Siebel database contains multiple repositories, then Siebel CRM gets all the field names from all repositories, and then displays them in the picklist. This configuration might result in Siebel CRM displaying duplicate values that you cannot choose in the picklist. To correct this situation, you create a search specification that configures Siebel CRM to read data only from the current repository.
3. Deliver your modifications.
4. Repeat Step 1 to make sure the Response Field drop-down list does not display duplicate values.

Restricting the Objects That Developers Can Modify

In some situations, it might be helpful to restrict the objects that developers can modify, according to a date that you specify. If you enable this feature, then the restrictions are applied when the developer does the following:

- Imports or exports a SIF or SDF file. For more information, see *Files That You Use to Manage Repositories*.
- Directly modifies the Siebel Database.

To restrict the objects that developers can modify

1. Select System Preferences from the Tools menu to navigate to the System Preferences view.
2. In the System Preferences list, set values for the system preferences shown in the following table.

System Preference Name	System Preference Value
EnablePerfFieldModification	Set the value to TRUE. This system preference applies edit and copy restrictions to objects.
Enable Object Version Control	Set the value to TRUE.
ServerEditRecordTimeStamp	Enter the time when Siebel must start using this configuration. For more information, see *Setting the Server Edit Record Time Stamp Parameter*.

3. For the changes to take effect, log out of the application and then log back in again.

Setting the Server Edit Record Time Stamp Parameter

When you set the ServerEditRecordTimeStamp parameter, you enter the time when Siebel starts using this configuration. Enter the time using the following format:

```
MM/DD/YYYY
```

You can specify only the month, day, and year. You cannot specify minutes and seconds.

For example, to start using the configuration on October 9, 2020, then enter the following value:

```
10/09/2020
```

Doing this:

- Allows you to modify any object that includes a timestamp that occurs after October 9, 2020.
- Does not allow you to modify any object that includes a timestamp that occurs before October 9, 2020. Developers can view or copy these objects, but not modify them.
- Allows you to create new records, and then to modify these new records.
- Allows you to copy any record, regardless of the timestamp.

Restricting Access to an Object

The following procedure describes how to restrict access to an object by configuring the ServerEditRecordTimeStamp parameter.

To restrict access to an object

1. Determine the date that the object was most recently updated.
   a. In the Object List Editor, choose the record where you want to restrict access.
   b. Click Help, and then click About Record.
   c. In the dialog box that opens, note the date in the Updated On field.
2. Set the value for ServerEditRecordTimeStamp to the date that you noted in Step 1c.

Properties That Affect Database Performance

The following table lists the properties that affect database performance.

Object Type	Property
business component	The following property affects database performance: Force Active.
field	The following properties affect database performance: • Force Active • Link Specification • Immediate Post Changes
multi value link	The following property affects database performance: Check No Match.
multi value field	The following properties affect database performance: • Force Active • Link Specification
single value field	The following properties affect database performance: • Force Active • Link Specification

Overriding Restrictions That Prevent Developers from Modifying Objects

This topic describes how the developer can modify any object, regardless of the timestamp restrictions that you place in Step1 of *Restricting the Objects That Developers Can Modify*.

To override restrictions that prevent developers from modifying objects

1. In Microsoft Windows, click Start, click All Programs, and then click for example the *Siebel Tools* installation folder.
2. Right-click the *Siebel Tools* icon, and then click Properties.

3. In the Properties pane, click the Shortcut tab, add the EditPerfFields switch to the Target field, and then click OK.

   For example:

   ```
   $SIEBEL_HOME\BIN\siebdev.exe /c "$SIEBEL_HOME\BIN\enu\tools.cfg"
   /u SADMIN /p ******** /d Sample / EditPerfFields
   ```

4. Log in to Siebel Tools.

Exporting and Importing Repositories

This topic describes how to export and import repositories. It includes the following information:

- *Overview of Exporting and Importing Repositories*
- *Importing Repositories in Windows Environments*
- *Importing Repositories in UNIX Environments*
- *Exporting Repositories*
- *Importing or Exporting Repositories at a Later Time*

Overview of Exporting and Importing Repositories

You can use the Database Configuration Wizard to export or import an entire repository – as detailed in the remainder of this chapter. For information on how to import objects from or export objects to an archive, see *Archiving Objects*.

Note: Whereas you use the Database Configuration Wizard to export or import entire repositories, you use the database utilities that your RDBMS vendor provides to back up the entire contents of the Siebel database.

For more detailed information about the Database Configuration Wizard, see *Siebel Database Upgrade Guide* and the *Siebel Installation Guide* .

Guidelines for Exporting and Importing Repositories

If you use the Database Configuration Wizard to import or export a repository, then it is recommended that you adhere to the following guidelines:

- Make sure to perform a Full Publish if you modify the repository. Make a backup copy of the repository file in case you want to compare it to the contents of the updated repository. If necessary, you can use this comparison to verify that the import is identical to the backup.
- Use the database utilities that your RDBMS vendor provides to back up the entire contents of the Siebel database.
- Use the Migrate option of the Database Configuration Wizard if you customize the source repository. For more information, see *Process of Migrating Repositories*.
- If you import a custom repository, then the Database Configuration Wizard will restore all the languages that are part of the predefined repository when you do the import. For example, if you archive repositories weekly, and if your development repository includes ENU and DEU, then the wizard includes ENU and DEU when it imports one of the archived repositories. For more information, see *About Predefined Objects*.

Supported Character Encoding to Export and Import Repositories

The following table describes the character encoding that Siebel CRM supports when it imports or exports a repository. These databases must use the same Siebel version.

Source Database	Target Database
Code Page	Code Page
Unicode	Unicode
Code Page	Unicode

Where the Database Configuration Wizard Saves Log Files

If you export a repository in a Windows or UNIX environment, and if you use the Export Repository option of the Database Configuration Wizard, then this wizard saves log files in following directories:

- `SIEBSRVR_ROOT\log\exprep\output`
- `SIEBSRVR_ROOT\log\exprep\state`

Exprep is the default process name for the exprep utility. You can modify this value.

Importing Repositories in Windows Environments

This topic describes how to import repositories in a Windows environment.

To import repositories in Windows environments

1. Make sure Siebel CRM supports the databases that you intend to use.

   For information about supported character encoding to export and import repositories, see *Overview of Exporting and Importing Repositories*.
2. In Microsoft Windows, navigate to Start Programs menu, Settings, Control Panel, and then click Services.
3. Stop all Siebel Servers.

   For more information, see *Siebel System Administration Guide* and the *Siebel Installation Guide* .
4. Click the Start Programs menu, Programs, Siebel Enterprise Server Configuration 8.0, and then click Database Server Configuration.
5. In the Database Configuration Wizard, enter information when this wizard prompts you, and then click Next to continue.
6. Choose Import Repository when the wizard prompts you to specify a database operation.
7. Specify the following items:

   - To import your custom 8.x repository
   - The location of where the custom CustRep.dat file resides

8. When the wizard displays the Configuration is Complete window, choose one of the following options, and then click Next:
   - **Yes Apply Configuration Changes Now.** The wizard saves the configuration information that you entered, and you can open the Siebel Upgrade Wizard in Step 10.
   - **No I Will Apply Configuration Changes Later.** The wizard saves the configuration information that you entered, but you cannot open the Siebel Upgrade Wizard in Step 10.
9. In the Configuration Parameter Review window, review the configuration that you entered. To modify a value, click Back to return to the window that includes the parameter you want to modify, modify the parameter, and then click Next.
10. When the wizard prompts you to run the configuration, click one of the following:
    - **No.** The wizard does not save the configuration information that you entered. You must enter the database configuration parameters again.
    - **Yes.** The wizard saves the configuration information that you entered.
11. Click OK.

    The Database Configuration Wizard does one of the following, depending on the choice that you make in Step 8:
    - **Apply now.** Opens the Siebel Upgrade Wizard, and then calls the SQL generator to create the SQL scripts.
    - **Apply later.** Saves the configuration information, but does not open the Siebel Upgrade Wizard. You can restart the configuration, and then run the Upgrade Wizard later. For more information, see *Importing or Exporting Repositories at a Later Time*.

Importing Repositories in UNIX Environments

This topic describes how to import repositories in a UNIX environment.

To import repositories in UNIX environments

1. Do Step 1 and Step 3 of *Importing Repositories in Windows Environments* .
2. Make sure `$SIEBEL_ROOT` is the current folder.
3. Run a script, depending on the following UNIX shell that you are using:
   - **Korn shell.** Run siebenv.sh.
   - **C shell.** Run siebenv.csh.
4. Make sure that the following environment variables use the following values:
   - **SIEBEL_ROOT.** This path must end in `siebsrvr` (for example: `/usr/siebel/siebsrvr/`).
   - **LANGUAGE.** Determines the language that the Database Configuration Wizard uses. The value of this variable is a text string that identifies the language (for example: `enu` for English).
5. Run the following command to start the Database Configuration Wizard:

```
$install_path/config/config -mode dbsrvr
```

6. In the Database Configuration Wizard, do the following:
   - Enter information when the wizard prompts you.
   - Use the Next and Back button to navigate between dialog boxes.

   - Choose Import Repository when the wizard prompts you to choose a database operation.
   - Specify to import your 8.x repository.
   - Identify the location of where the custom CustRep.dat file resides.
7. After you enter all the requested information, the wizard displays a message that is similar to the following. Click Next to continue:

```
Configuration is complete: configuration parameters will be saved to $Masterfile
file when the wizard completes. Please run the following command line after you
exit from this configuration wizard. This command will deploy the process you
configured to the database.

$SIEBEL_ROOT/siebsrvr/bin/srvrupgwiz /m $SIEBEL_ROOT/siebsrvr/bin/$Masterfile
```

8. The wizard displays the values that you entered in the Parameter Review window. To modify a value, click Back to return to the appropriate window.
9. The wizard prompts you to click one of the following values:
   - **Yes**. The wizard saves the configuration in a master file in the `$SIEBEL_ROOT/bin` folder. It does not start the Upgrade Wizard. For information about how to start the Upgrade Wizard, see *Siebel Database Upgrade Guide* .
   - **No.** The wizard does not save the configuration that you entered.

Exporting Repositories

To export a repository when you use:

- **Microsoft Windows.** Use the same procedure that you use to import a repository, but choose Export Repository in Step 6 of *Importing Repositories in Windows Environments*.
- **UNIX.** Use the same procedure that you use to import a repository, but choose Export Repository in in Step 6 of *Importing Repositories in UNIX Environments*.

Importing or Exporting Repositories at a Later Time

If you use the Database Configuration Wizard to export or import a repository, then it saves the values that you specify to the master_exprep.ucf file in the `SIEBSRVR_ROOT\` folder. After the wizard finishes collecting information from you, it prompts you to export or to not export. If you choose not to export or not to import, then you can run the following command to do the export or import at a later time:

```
siebupg.exe /m master_exprep.ucfs
```

Upgrading Repositories

The *Siebel Application Upgrader* is a utility that you can use to get new features from the latest software release while preserving the custom configuration that you created in the current repository. It allows you to do the following:

- Compare your custom configuration to the modifications that a new Siebel CRM release contains.
- Receive notifications about conflicts between the customizations that you make and the new release.

- Merge any differences between objects.
- Choose the modifications to apply and manually override modifications.
- Merge objects with versions, including task UI objects and Workflow Processes. It copies version 1 through version `n` from the prior custom repository to the new custom repository. It merges version 0 from the prior repository with the new custom repository. This configuration results in version `n+1` in the new custom repository.
- Reduce the time required to upgrade Siebel CRM.

You can use the Application Upgrader to merge an entire custom repository with another repository. To merge only part of a repository, you must import the repository. For more information, see *Exporting and Importing Repositories*. For more information about the Application Upgrader, see *Siebel Database Upgrade Guide* .

To upgrade repositories (Siebel Tools)

1. Click the Tools menu, click Upgrade, and then click Upgrade Application.
2. In the Merge Repositories dialog box, choose the repositories to merge, and then click Merge.

   The following then happens:

   - The upgrade starts.
   - Any differences between objects and properties are displayed.
   - Any differences between objects and properties for different versions of task UIs and Workflow Processes are displayed.

Identifying Conflicts That Occur During Upgrades

This topic describes how to identify merge conflicts between objects that are added or modified during a repository merge. A *merge conflict* is a scenario in which a customer changed an object in the current customer repository and Siebel CRM changed that same object to a different value in a new version of the Siebel Repository. Consequently, the attribute for the object is different when the following three repositories are compared: the current customer repository, the current Siebel Repository, and the new version of the Siebel Repository.

To identify conflicts that occur during upgrades (Siebel Tools)

1. Make sure you finish upgrading repositories.

   For more information, see *Upgrading Repositories*.
2. Click the Screens menu, click Application Upgrader, and then click Application Upgrade Object List.
3. In the Application Upgrades list, right-click the record of the merge that you want to analyze, and then click Hierarchy Reports.

   The hierarchy report appears, which includes the objects that were added or modified during a merge. The report includes the following types of objects:

   - **Objects that include a valid value for each field value.** These objects were modified during the merge. An N or a Y in a binary field is an example of a valid value.

     For these objects, the Status field designates added objects or objects with modified attributes. For added objects, an N appears in the Attributes field because none of the attributes are modified. For objects with modified attributes, a Y appears in the Attributes field, and the modified attributes appear in the Attributes pane.

  - **Objects that include an asterisk (*) for each field value.** Children of these objects were modified during the merge, or the dependent objects of these objects were modified during the merge.

4. To view information about merge details, complete the following steps:
    a. Expand the (Object Types) hierarchical tree and select a parent or child object type.
    b. Select an object in the List of Objects list.

       In the Attributes pane, the merge conflicts for the selected object are listed.
5. (Optional) To filter the objects in the report, select one of the following values in the Filter drop-down list, and then click Go:
    - **Siebel and Customer Modified.** Displays only objects that Oracle or you modified. These objects have a Y in the In Prior Standard, In Prior Customized, and In New Standard fields.
    - **Siebel Modified.** Displays only objects that Oracle modified or added. These objects have an N in the In Prior Standard and In Prior Customized field and a Y in the In New Standard field.
    - **All.** Displays all modified or added objects.
6. (Optional) To filter the merge conflicts in the Attributes pane, select or clear the Critical Only check box as follows, and then click Go:
    - Select the check box to show only critical merge conflicts.
    - Clear the check box to show all merge conflicts.
7. (Optional) To display the dependencies for an object, complete the following steps:
    a. Click an object in a List of Objects pane.

       If you want to display the dependencies for an additional object, then hold down the CTRL key, and click the additional object in the List of Objects pane.
    b. Click Show Dependencies (if the object has no dependencies, then the Show Dependencies button is disabled).

       The dependent objects that were modified are displayed and the attributes for these objects.
    c. Click Back to return to the hierarchy report.

Process of Migrating Repositories

Configuration and other changes must be migrated from your development environment to your downstream environments, such as test or production. To perform this migration, use the Siebel Migration Application. For more information, see *Siebel Database Upgrade Guide*.

Deleting Repositories

You can delete repositories using either the repimexp utility or the RRCleanup utility.

- **repimexp utility.** Using this utility, you can delete the existing repository using the `/a t` option.
- **RRCleanup utility.** Using this utility, you can delete the existing repository or the orphan records from Siebel database as follows:

- **Repository data.** This option passes a valid repository as an argument to the RRCleanup utility to delete repository data from all repository tables. The RRCleanup utility deletes the LOVs (S_LST_OF_VAL) as well for this repository.
- **Orphan records.** This option does not require you to pass a repository name as an argument to the RRCleanup utility because orphan records do not have the valid repository in the Siebel system and the repository does not exist in the S_REPOSITORY table. The RRCleanup utility deletes the LOVs (S_LST_OF_VAL) as well for orphan Workspaces.

Note: A repository can contain multiple integration branches with each branch containing a copy of List of Value. Deleting a repository has a cascading effect on branches and LOVs.

The RRCleanup utility is located at this location (in Siebel Tools):

```
<Siebel_Home>\BIN\RRCleanup.exe
```

The RRCleanup utility is located at this location in the Siebel Server:

```
<Siebel_Home>\ses\siebsrvr\BIN\RRCleanup.exe
```

The following procedure shows how to run the RRCleanup Utility.

To run the RRCleanup utility

1. Open the Command Prompt window from your computer by clicking the Start button and then selecting the Run option.

   The Run window appears.
2. Enter the value CMD in the Open field.

   The Command Prompt window appears.
3. Change the directory in the Command Prompt window using one of the following commands:

   ```
   cd <Siebel_Home>\BIN
   ```

   or

   ```
   cd <Siebel_Home>\ses\siebsrvr\BIN
   ```
4. Run the RRCleanup.exe utility.

5. The Command Prompt window opens and displays the following arguments and parameters, which you can use to run the RRCleanup utility:

    - `-t` Siebel Table Owner (required)
    - `-u` Username (required)
    - `-p` Password (required)
    - `-o` ODBC Data Source (required)
    - `-l` Log File Name (default: RRCleanup.log)
    - `-r` Repository Name (delete Runtime Repository data for repository name pass. For example: “Siebel Repository”)
    - `-a` Orphan Flag (Pass Y if required to delete only orphan records)
    - `-s Siebsrvr/Tools Installation` path specified (required)
    - `-b` Repository Type ((R)unTime, (B)Both (DR and RR)) (required)
    - `-d` DB Platform Name (Oracle, MSSQL, DB2UDB, or DB2390)

6. Use the following command to delete a particular repository data from repository tables:

    ```
    RRCleanup.exe -t "Table Owner" -u "Siebel User Name" -p "Siebel User Name Password"
    -r "Repository Name" -o "ODBC Data Source" -s "Siebsrvr/Tools home location" -b Repository Type
    ```

    Note: For the `-r` argument, you can pass only one repository at a time as a parameter. For the `-b` argument, you can pass either of the repository types as parameter such as `-b R` to delete data from Runtime Repository tables or `-b B` to delete data from all repository tables.

7. Use the following command to delete orphan records:

    ```
    RRCleanup.exe -t "Table Owner" -u "Siebel User Name" -p "Siebel User Name
    Password" -a Y -o "ODBC Data Source" -s "Siebsrvr/Tools home location" -b Repository Type
    ```

    Note: In one command, you can delete either a repository or orphan records but not both. However, if you pass a repository as parameter and the orphan flag is also set to true then the orphan takes precedence.

8. View the output stages that represent the progress of the tasks performed by the utility to ensure that the RRCleanup utility is run successfully.

About Removing Repository Data and/or Orphan Records

Before running the RRCleanup utility, note that the following options are available to remove Runtime Repository (RR) data and/or LOV orphan records:

1. **Option 1,** `-b R`: Removes all RR definitions only.

    In a RR environment, all RR definitions will be removed but LOVs will remain. You could run this option before a Full Publish operation in a Design Time Repository (DR) environment (however, it is not necessary since Full Publish will do this automatically). For example, the following removes the RR "RepName" :

    ```
    -r "RepName" -b R -a N <-- (or just leave out -a N) -->
    ```

2. **Option 2,** `-b B`: Removes all RR and DR definitions for Workspace-enabled objects and LOVs. You should use this option in RR environments. For example, the following removes the DR, RR, and LOVs associated with "RepName" :

   ```
   -r "RepName" -b B -a N <-- (or just leave out -a N) -->
   ```

   This is the best option.

   > **Tip:** If you have orphaned data because you ran `-b R` for example instead of `-b B`, then using `-a` (without any `-b`) will get rid of that.

3. **Option 3,** `-b B a Y`: Removes any orphan data in DR, RR, or LOVs – that is, all records with a WS_ID that do not exist in S_WORKSPACE. If you use Option 2 all the time, then you would never have to do this.

   ```
   -b B -a Y
   ```

10 Localizing Strings and Locale Data

Localizing Strings and Locale Data

This chapter describes how to localize strings and locale data. It includes the following topics:

Configuring Symbolic Strings

This topic describes how to configure symbolic strings. It includes the following information:

Overview of Configuring Symbolic Strings

A *symbolic string* is an object that you can use to store the value of a string. It allows you to define a string one time, and then reference it from multiple objects. An object can reference this symbolic string to get a literal value, and then display it as text in the Siebel client. For example, an applet can reference a symbolic string to get the text it shows in a control caption, a list column display name, or an applet title.

A *translatable string* is a type of string that Siebel CRM can translate to another language. For example, it can translate the text string that defines a control label from English to French.

The symbolic string centralizes all the strings in the repository. It includes strings in English and all other languages. It includes the following advantages:

- Reduces redundancy because many objects can reference one symbolic string
- Results in a more consistent user interface
- Simplifies maintenance because you are required to maintain only one string for a word or phrase
- Eliminates duplicate translations of the same word
- Reduces translation costs

Siebel CRM cannot translate a nontranslatable property to another language. For example, the HTML Sequence property, HTML Height property, and HTML Width property are nontranslatable properties.

If you deliver to the repository, then the current language mode that you set for Siebel Tools or Web Tools is used. It uses this language mode to choose from one of several translations for a set of symbolic string locale records. For more information on how to deliver your Workspaces, see *Delivering Workspaces*.

Symbolic Strings in the Object Hierarchy

The Symbolic String object type is a top-level object type. it includes the child Symbolic String Locale object type. Each symbolic string represents a word or phrase that is language independent. For example: Account

Siebel CRM stores all translations of this word or phrase, including English, as a symbolic string locale. For information about top-level object types, see *Displaying Object Types in the Object Explorer*.

Data for a symbolic string is stored in the S_SYM_STR (symbolic strings) table. Data for a symbolic string locale is stored in the S_SYM_STR_INTL (Repository Symbolic String Locale) table. Objects that reference symbolic strings store foreign key references to these strings in the S_SYM_STR table.

Symbolic strings do not include other types of strings that Siebel CRM typically includes as seed data, such as LOVs, error messages, or predefined queries. For more information about localizing these types of strings, see *Fixing Orphaned String References After an Upgrade*.

How a Translatable String is Determined

To compile an object property that displays a translatable string, such as the Title property of an applet, the following happens:

- If a value exists in the string language that the compile uses, then it compiles this string override value.
- If the string override field for the current language mode that is set does not include a string value, then the current language mode and the String Value property of the associated symbolic string locale are used to determine the value.

In most situations, a string override does not exist. For more information, see *Entering a String Override*.

How Some Early Releases Store Translatable Strings

Some early Siebel releases store translatable strings in the locale objects of a parent object type. For example, each applet includes a set of child locale records that define the text for the applet title that Siebel CRM shows in the Siebel client.

Modifying the Configuration File to Support Symbolic Strings

This topic describes how to modify the configuration file to support symbolic strings.

To modify the configuration file to support symbolic strings

1. Open the tools.cfg file.
2. In the Siebel section, set the EnableToolsConstrain parameter to one of the following values:

   a. **TRUE. Constrained mode.** This mode requires you to choose a translatable string from the list of available string references. You cannot override the string reference. For example, you cannot enter a value for a string override field. You cannot create a new symbolic string reference.

   b. **FALSE. Unconstrained mode.** This mode does not require you to choose a translatable string from the list of string references. You can override the string reference. For example, you can enter a value in a string override field. You can create a new symbolic string reference.

3. (Optional) Set the SymStrPrefix parameter to a custom value.

   To distinguish between a predefined symbolic string and a custom symbolic string that you create, Siebel sets the prefix for any new symbolic string you create to `x_`, by default. You can modify this value. For example, if you set this parameter to `SymStrPrefix=X_NewString`, then any new symbolic string you create includes the `X_NewString` prefix. For more information, see *Modifying a Predefined Symbolic String* and *About Predefined Objects*.

Modifying a Predefined Symbolic String

You can modify a *predefined symbolic string*, which is a string that comes predefined with Siebel CRM. It includes a `SBL_` prefix in the Name property. For more information, see *About Predefined Objects*.

CAUTION: It is recommended that you do not modify a predefined symbolic string. Instead, it is recommended that you create a new symbolic string that includes the text you require. It is recommended that you do not modify the display value of a predefined symbolic string. Siebel CRM might use this display value throughout the Siebel client. For a monolingual deployment, you risk modifying text in the Siebel client that you do not intend to modify. For a multilingual deployment, you risk breaking the relationships between display values for different languages. For more information, see *Creating a New Symbolic String*.

To modify a predefined symbolic string

1. In the Object Explorer, select the Symbolic String object type.

   For more information, see *Displaying Object Types in the Object Explorer*.
2. In the Symbolic Strings list, select the Symbolic String record that you want to modify.

   For more information, see *Locating Object Definitions in the Object List Editor*.
3. Set the Project property of the string you located in step 2 to the project you create.

   To modify multiple records, see *Modifying Objects*.

Creating a New Symbolic String

The following procedure describes how to create a new symbolic string.

To create a new symbolic string

1. Make sure the EnableToolsConstrain parameter allows you to create a new symbolic string.

   For more information, see *Modifying the Configuration File to Support Symbolic Strings*.
2. In the Object Explorer, select the Symbolic String object type.

   For more information, see *Displaying Object Types in the Object Explorer*.
3. In the Symbolic Strings list, create a new record with the values shown in the following table.

Property	Description
Name	Enter a unique name for this symbolic string. For more information about the prefix that is added to the name of any custom symbolic string you create, see *Modifying the Configuration File to Support Symbolic Strings*.
String Value	Enter the text that this symbolic string shows. In some situations, this value might be a value that is determined according to the current language mode and the String Value property of the child symbolic string locale object. For more information, see *Setting the Language Mode*. Trailing spaces are truncated, including full width spaces in Japanese.
Definition	Enter text that describes the purpose of this symbolic string.
Project	Set this property to the project you create.

Modifying a Symbolic String to Globally Update Display Values

You can configure Siebel CRM to make global modifications to the values it shows in the Siebel client. For example, this feature can be useful in the following situations:

- You want to modify all instances of the word Account that Siebel CRM shows in the Siebel client to Customer.
- You want to deploy Siebel CRM specific to an industry in a locale other than English. The text strings that it shows in the Siebel client might not be appropriate for this industry.

To modify a symbolic string to globally update display values

1. Set the language mode.

   For more information, see *Setting the Language Mode*.
2. In the Object Explorer, select Symbolic String, then the Symbolic String Locale object type.

   For more information, see *Displaying Object Types in the Object Explorer*.
3. In the Symbolic Strings list, select the Symbolic String record that you want to modify.
4. Modify the value of the String Value property.
5. Deliver the projects that include the objects that reference this symbolic string.

Setting a Symbolic String Reference

A *symbolic string reference* is a reference to a symbolic string. It allows you to set the text that Siebel CRM shows in the Siebel client for an object property, such as an applet title or application display name. This topic describes two different ways that you can set this text.

Using the String Reference Property to Set a Symbolic String Reference

This topic describes how to use the String Reference property to set a symbolic string reference.

To use the String Reference property to set a symbolic string reference

1. Make sure the EnableToolsConstrain parameter is set to FALSE.

   For more information, see *Modifying the Configuration File to Support Symbolic Strings*.
2. In the Object List Editor, locate the object you want to modify.

   For example, locate the Account List Applet. For more information, see *Locating Object Definitions in the Object List Editor*.
3. In the Title - String Reference property, click the drop-down arrow.

   This property varies according to object type.
4. In the (Title - String Reference) dialog box that opens, locate the string reference you require, and then click OK.

   This populates the Title property according to the current language mode and the value in the String Value property of the child symbolic string locale.
5. If no existing symbolic string meets your requirements, then you can enter a string override.

   For more information, see *Entering a String Override*.

Properties That Store a Symbolic String

The following table describes the properties that store a symbolic string. The property varies according to object type.

Object Type	**Label for the String Reference Property**
Web Page	Title - String Reference
Applet	Title - String Reference
Screen	Viewbar Text - String Reference
View	A view includes the following properties that reference a symbolic string: • Title - String Reference • Thread Title - String Reference
Application	Display Name - String Reference

Entering a Value to Set a Symbolic String Reference

This topic describes how to enter a value to set a symbolic string reference.

To enter a value to set a symbolic string reference

1. Make sure the EnableToolsConstrain parameter is set to FALSE.

   For more information, see *Modifying the Configuration File to Support Symbolic Strings*.
2. In the Object List Editor, locate the object you want to modify.

   For example, locate the Account List Applet. For more information, see *Locating Object Definitions in the Object List Editor*.
3. In the Title property, enter the text that Siebel CRM must show in the Siebel client.

   This property varies according to object type. .
4. Tab out of the field.

   Siebel searches for a string reference that includes a value in the String Value property that matches the value you enter, and then does one of the following depending on the search result:

   - **A unique match exists.** It enters the symbolic string reference it finds into the String Reference property. It enters this value according to the current language mode and the value that the String Value property of the child symbolic string locale contains.
   - **Multiple matches exist or no match exists.** It displays an error dialog box. Click OK to close the dialog box, and then use the String Reference Property. For more information, see *Using the String Reference Property to Set a Symbolic String Reference*.

Properties That Store the Display Text

The following table describes the properties that store the display text. The property varies according to object type.

Object Type	Property That Stores the Display Text
Web Page	Title
Applet	Title
Screen	Viewbar Text
View	A view includes the following properties that include display text: • Title • Thread Title
Application	Display Name

Entering a String Override

If you cannot find a symbolic string that meets your requirements, then you can override the symbolic string that Siebel CRM uses. The information in the Properties That Store the Display Text section of *Entering a Value to Set a Symbolic String Reference* lists the properties that store a translatable string. Siebel CRM includes a corresponding String Override property for each of these properties that store a translatable string. For example, the Title – String Override property is the corresponding property for the Title property of an applet. You can use this override property to enter a custom string.

To enter a string override

1. Make sure the EnableToolsConstrain parameter is set to FALSE.

   For more information, see *Modifying the Configuration File to Support Symbolic Strings*.
2. In the Object List Editor, locate the object you want to modify.

   For example, locate the Account List Applet. For more information, see *Locating Object Definitions in the Object List Editor*.
3. In the Title - String Override property, enter the string you require, and then tab out of this field.

   This property varies according to object type. For more information, see *Properties That Store the String Override*.

   For more information, see *How the Override Value You Enter is Stored*.

How the Override Value You Enter is Stored

If you modify the value in the Title-String Override property, then Siebel also modifies the value in the Title property to the custom value you enter. It stores this value in a locale object that is a child of the parent you modify.

For example, if you use the Title-String Override property to modify the display name for the Account List Applet to My Big Accounts, then Siebel adds a child locale object to the Account List Applet. The Title property of this locale object includes the custom value you enter, such as My Big Accounts. To view this locale object, you must display it in the Object Explorer. For more information, see *Displaying Object Types in the Object Explorer*.

To set this value, Siebel uses the current language mode and stores it as a language-dependent value. This value does not affect other references to symbolic strings.

Properties That Store the String Override

The following table describes the properties that store the string override. This property varies according to object type. These properties appear in the Object List Editor and in the Properties pane.

Object Type	**Property That Stores the String Override**
Web Page	Title - String Override
Applet	Title - String Override
Screen	Viewbar Text - String Override
View	A view includes the following properties that store a string override: • Title - String Override • Thread Title - String Override
Application	Display Name - String Override

Object Type	Property That Stores the String Override

Converting Symbolic Strings

This topic describes how to convert symbolic strings. It includes the following information:

- *Consolidating Symbolic Strings*
- *Using a Batch File to Convert Strings*

You can convert a translatable string to a symbolic string. Siebel CRM stores this translatable string as a child locale record of a top-level object type. It stores symbolic strings in a single table. Converting a translatable string might be useful in the following situations:

- You upgrade to a new Siebel CRM version and you want to convert custom translatable strings to symbolic strings.
- You use string overrides to store text strings and periodically want to convert them to symbolic strings.

Converting to symbolic strings reduces the size of the repository, simplifies translations, and helps to make sure that the text that Siebel CRM shows in the Siebel client is consistent. It also requires that development work is finished.

To convert symbolic strings

1. Review the caution information described in *Caution About Converting and Consolidating Symbolic Strings*.
2. Prepare to do the conversion:

   a. Back up the Siebel database and repository.
   b. Verify the configuration file:

      - Make sure the DataSource parameter references the correct Siebel database. The conversion utility uses this Siebel database.
      - Make sure the EnableToolsConstrain parameter is set to FALSE and that the SymStrPrefix parameter is set to the appropriate prefix.

      For more information, see *Modifying the Configuration File to Support Symbolic Strings*.
   c. Make sure all projects are unlocked.
   d. Inform other developers not to log on to the development environment while the conversion runs.
3. In the Object Explorer, select the Attribute object type.

   For more information, see *Displaying Object Types in the Object Explorer*.

4. Identify the object types you want to convert:

    a. In the Object Explorer, click the Flat Tab, and then click Attribute.
    b. In the Attributes list, query the Name property for the following string:

    ```
    *String Reference*
    ```

    The Parent Type property displays the complete set of object types to convert. If an object type includes more than one property that references a symbolic string, then you can run the conversion for this object type only one time.

5. Convert locale strings to symbolic strings. You must convert the locale strings for the object types you identify in Step 4. You can use the conversion utility to do this conversion:

    a. Open a command line, and navigate to the `$SIEBEL_HOME\BIN` folder.
    b. Enter the following command:

    ```
    consoleapp config_file app_lang user_Id password "business_service"
    "business_service_method: parameters"
    ```

    The following table describes the parameters in the command that you can set. For more information, see *Running the Console Application Executable*.

Parameter	Description
config_file	Specify the configuration file, such as tools.cfg. Use the default data source. For more information, see *Modifying the Configuration File to Support Symbolic Strings*.
app_lang	Specify the application language, such as ENU.
user_Id	Specify the user Id.
password	Specify the password.
business_service	Specify the String Conversion business service.
business_service_method	Specify the name of the business service method.
parameters	Specify the input parameters to the business service method. For more information, see *Separating Conversion Files into Smaller Files*.

6. Export candidates for one of the object types you identified in Step 4. Enter the following command:

    ```
    consoleapp "ConversionExport: Filename=name.txt, Repository=repository_name,
    Object=object_type, LogFile=object_typeExport.log, Language=language_code,MatchMin=an_ineger"
    ```

    For example, the following command exports the control object type:

```
consoleapp "ConversionExport: Filename=Control.txt,Repository=Siebel Repository,
 Object=Control, LogFile=ControlExport.log, Language=ENU, MatchMin=1"
```

For information about setting parameters in this command, see *Conversion Export Utility Parameters*.

For more information, see *How the Conversion Export Utility Converts Strings*.

7. Repeat Step 6 for each object type that you identified in Step 4.
8. Import the symbolic strings that you converted in step 6. Enter the following command:

```
consoleapp "ConversionImport: Filename=name.txt, Repository=repository_name,
LogFile=name.log, UnlockProjects=false, SkipParentUpdates=true, Project=Symbolic Strings"
```

For example:

```
consoleapp "ConversionImport: Filename=Control.txt, Repository=Siebel
Repository, LogFile=ConversionImport.log, UnlockProjects=false, SkipParentUpdates=true,
Project=Symbolic Strings"
```

For information about setting parameters in this command, see *Conversion Import Utility Parameters*.

For more information, see *How the Conversion Import Utility Converts Strings*.

9. Consolidate strings.

For more information, see *Consolidating Symbolic Strings*.

Caution About Converting and Consolidating Symbolic Strings

Converting or consolidating symbolic strings might consume computer processing resources.

CAUTION: Converting or consolidating symbolic strings might consume a significant amount of computer processing resources. For information about the requirements for computer processing speed, see *Siebel System Requirements and Supported Platforms* on Oracle Technology Network.

Note: For Siebel CRM product releases 8.1.1.9 and later and for 8.2.2.2 and later, the system requirements and supported platform certifications are available from the Certifications tab on My Oracle Support.

File and Object command-line parameters are case sensitive.

CAUTION: File and Object command-line parameters that you use when you convert or consolidate a symbolic string are case sensitive. Other command-line parameters are not case sensitive.

Running the Console Application Executable

You can use consoleapp.exe (console application executable) to run various utilities that you use to manage symbolic strings.

To run the console application executable

1. On the computer where you installed your Siebel application, go to the `$SIEBEL_HOME\BIN` folder.
2. Enter the following command:

   ```
   consoleapp parameter_1 parameter_n
   ```

   This command runs the console application executable.

Separating Conversion Files into Smaller Files

The command that this topic describes separates an export file that the conversion utility creates into smaller, more manageable files according to object type. For example, the Siebel Runtime Repository might include up to 130,000 controls. To improve performance, you can simultaneously import multiple consolidation files of the same object type or of different object types. An average desktop PC can typically run only 10 simultaneous conversion import processes.

To separate conversion files into smaller files

- Add the following parameter to the command you used in Step 5 of *Converting Symbolic Strings*:

  ```
  "SplitFile: Filename=Control.txt, Lines=2000"
  ```

  `Filename` is the name of the file to export.

  `Lines` is the approximate number of lines in each file. The conversion utility does not separate a set of symbolic strings, so the number of lines might not equal the value you set in this parameter.

How the Conversion Export Utility Converts Strings

The Conversion Export utility can convert any object that includes a translatable string (for example: a control, list column, or applet). The export utility does the following work:

1. Creates a sorted list of English (ENU) child records for each translatable string in an object.
2. Sequentially processes each object that includes multiple translatable strings. For example, a list column that includes a display name and prompt text includes multiple translatable strings.
3. Uses the list it creates in Step 1 to create information about any new symbolic strings.
4. For records that include identical ENU translations, it compares the non-ENU records and reuses the same symbolic string for subsequent records, if possible.
5. Repeats Step 1 through Step 4 for the next object type.
6. Produces an output file that includes information about the new symbolic strings, including information about each language translation and replacement strings. This file is for information purposes. It is not a log file. It is not necessary to review the contents of this file.

How the Conversion Import Utility Converts Strings

To convert the records that use the new symbolic strings, you can use a utility that performs inserts, updates, and deletes on the Siebel database. For input, this utility uses the output file that the conversion export creates. This utility does the following:

1. Creates a new symbolic string record and child symbolic string locale records for each string according to the string values in the target objects. The export file includes information about each string, including a unique name and information about each child locale object.
2. Adds a reference to the new symbolic string in the relevant property of each original object. For example, 10 applets exist and the title for each of these applets is My Big Service Requests. The non-ENU values for these titles all use the same value. In this example, the export file includes information about one new symbolic string and instructions for each of the 10 applets to use this new symbolic string as the title. The conversion import does the following:
    - Creates a symbolic string with an ENU value of My Big Service Requests.
    - Sets the Title - String Reference property for each of the 10 applets to the name of this new symbolic string.
    - Clears redundant values for child locale objects. Each of the 10 applets now include a value in the String Reference property and a value in the String Override property. The value in the String Override property is redundant and the conversion import clears this value for each child locale object.
3. Deletes any records that the object locale records no longer reference.

For more information, see *Converting Symbolic Strings*.

How Siebel CRM References Symbolic Strings at Runtime

Siebel CRM uses the same repository before and after a conversion. For example, an applet gets the value for the Title property from a child applet locale record. During conversion, Siebel does the following:

1. Creates a symbolic string.
2. Places the reference for this symbolic string in the Title - String Reference property of the applet.
3. Removes the applet locale record.

After Siebel finishes conversion it gets the applet title from the symbolic string. The display value for the title that it compiles is the same value that this title contained before the conversion. Siebel compiles strings into object definitions in the repository. Siebel CRM reads symbolic strings from these object definitions. It does not read them from the S_SYM_STR table at runtime.

Consolidating Symbolic Strings

Consolidating symbolic strings eliminates duplicate symbolic strings that the conversion utility might create when it converts these strings. The conversion utility converts all the symbolic strings for one object type, and then converts all the symbolic strings for the next object type. It does this until it finishes converting all the symbolic strings that you specify. It creates multiple symbolic string records for the same display value that occurs in multiple object types. Duplicate symbolic strings can include identical sets of locale records or one symbolic string might include more child locale records than another string. The strings that these objects use in common are identical. Consolidation can remove these duplicates.

To consolidate symbolic strings

1. Review the important caution information described in *Caution About Converting and Consolidating Symbolic Strings*.
2. Convert symbolic strings.

   For more information, see *Converting Symbolic Strings*.
3. Run the following command to merge duplicate symbolic strings:

   ```
   consoleapp "ConsolidationExport:Filename=ConsExp.txt,Repository=Siebel
   Repository,LogFile=ConsolidationLog.txt,Language=ENU,MatchMin=2"
   ```

   The following table describes the parameters in the command that you can set. For more information, see *Running the Console Application Executable*.

Parameter	Description
Filename	The name of the export file.
Repository	The name of the repository.
LogFile	The name of the log file.
Language	Same as the Language parameter that you use to export candidates for an object type. For more information, see *Separating Conversion Files into Smaller Files*.
MatchMin	The minimum number of matches that the utility must find in a set of matching symbolic strings before it writes them to the file. The default value is 2.
SkipSBLStrings	You can use one of the following values: ○ **TRUE.** Ignore all strings that include the SBL_ prefix in the Name property. ○ **FALSE.** Consolidate all strings that include the SBL_ prefix and all custom strings that you create. ○ **MasterOnly.** Do not consolidate strings that include the SBL_ prefix, but use them as master strings. The default value is TRUE.

   After you convert locale strings to symbolic strings, you can use the consolidation utility to find duplicate symbolic strings and merge them and their references into a single symbolic string. As input, this utility uses the file that the consolidation export creates. It deletes redundant symbolic strings. It replaces all references that objects make to these strings with a reference to the master string. This consolidation might consume a significant amount of time because the object types in the repository include approximately 80 translatable string properties.
4. Import consolidated strings using the ConsolidationImport business service method.

For example:

```
consoleapp "ConsolidationImport:Filename=ConsExp.txt,Repository=Siebel
Repository, LogFile=ConsolidationLog.txt,UnlockProjects=false,SkipParentUpdates=true"
```

These parameters are the same parameters that you use to import the symbolic strings in Step 8 of *Converting Symbolic Strings*. For more information, see *Running the Console Application Executable*.

5. (Optional) Separate a consolidation export file into smaller files.

   To separate a consolidation export file into smaller files, you use the same parameters that you use to separate a conversion export file. You can then do Step 4 to import each of these smaller files. For more information, see *Separating Conversion Files into Smaller Files*.

Using a Batch File to Consolidate Strings

You can use a batch file to consolidate strings.

To use a batch file to consolidate strings

1. Navigate to the `$SIEBEL_HOME\BIN` folder.
2. Enter the following command:

   ```
   strcons Action User_Id Password
   ```

   You use the same parameters that you use when you use a batch file to convert strings. The only difference are:

   - If you set the action parameter to Import, then the utility imports the files from the working folder that the TEST_LOCATION parameter in the batch file identifies.
   - You do not specify an object type.

   For more information, see *Using a Batch File to Convert Strings*.

Using a Batch File to Convert Strings

You can run the conversion utility from a batch file.

CAUTION: If the Siebel installation path includes a space, then you must enclose this path in quotes.

To use a batch file to convert strings

1. Navigate to the `$SIEBEL_HOME\BIN` folder.

2. Enter the following command:

```
strconv "object_type" action user_Id password
```

For example:

```
strconv "Applet" export user_Id password
```

The following table describes the parameters you can use to run the strconv.bat (string conversion) batch file. You set other parameters in the batch file. For information about the batch file, see comments that the batch file contains.

Parameter	Description
object_type	Object type to convert. For example: ◦ `Applet` ◦ `Control` ◦ `List Column`
action	You can use one of the following values: ◦ **Export.** The utility exports all convertible locale records. ◦ **Import.** The utility imports the file that the object_type parameter identifies.
user_Id	The user name that you use to log in to the Siebel application.
password	The user password that you use to log in to the Siebel application.

Configuring Locale Data

This topic describes how to configure locale data. It includes the following information:

- *Configuring Nontranslatable Locale Object Properties*
- *Displaying Controls and List Columns According to Locale*
- *Fixing Orphaned String References After an Upgrade*

Configuring Nontranslatable Locale Object Properties

A locale can be one of the following:

- **Translatable.** For example, a text string that defines a control label.
- **Nontranslatable.** For example, the HTML Sequence property, HTML Height property, or HTML Width property of a control.

User interface conventions can vary by locale. For example, one locale might require a different sequence of fields from another locale. To configure a nontranslatable object property for a locale, you can enable language override mode. This mode allows you to store a nontranslatable, locale property as a child locale record of the parent object.

In the following example, a Siebel enterprise requires Japanese (JPN) and four Western European languages. Japanese does not use middle names but Western European languages do use middle names. Japanese uses the family name first. Western European languages use the family name last.

To configure nontranslatable locale object properties

1. Set the language mode to JPN.

   For more information, see *Setting the Language Mode*.
2. Enable language override.

   If you do not enable language override, and if you compile any of the Western European languages, then Siebel CRM uses the Japanese configuration of no middle name and family name first. For more information, see *Setting the Language Mode*.
3. In the Object List Editor, locate the object you want to modify.

   For more information, see *Locating Object Definitions in the Object List Editor*.
4. To define locale values, modify the object properties:
   a. To hide the middle name, remove the value from the Title-String Override property.
   b. Reverse the order of the first name and last name.

      You modify these locale values to meet the needs that this example requires. The locale values you must modify depend on your business requirements. You can use the Object List Editor or the Siebel IDE editors.
5. Deploy your changes to the Siebel Runtime Repository.

Displaying Controls and List Columns According to Locale

You can control how Siebel CRM displays a control or list column according to the locale requirements.

CAUTION: To hide or display a control or list column, it is recommended that you modify a property. If you delete a control or list column, then Siebel deletes it from all languages even if you enable language override.

The following example displays locale objects in an applet.

To display controls and list columns according to locale

1. Set the language mode.

   For more information, see *Setting the Language Mode*.
2. In the Object Explorer, expand the Applet tree.
3. Click Control or Expand the List tree and click List Column.
4. In the Object List Editor, locate the object you want to modify.

   For more information, see *Locating Object Definitions in the Object List Editor*.
5. Modify the properties for the selected object as required.

The following table describes some of the values you can modify.

Object Type	Property	Description
Control	Visible	If TRUE, then Siebel CRM displays this control in the Siebel client in the parent language and in all other languages that it supports.
Control	Visible-Language Override	If you enable language override, then you can set this property to one of the following values: ◦ **FALSE.** Hides the control in the Siebel client. ◦ **TRUE.** Displays the control in the Siebel client. For more information, see *Setting the Language Mode*.
List Column	Show in List	If TRUE, then Siebel CRM displays this column in the Siebel client in the parent language and in all other languages that it supports.
List Column	Show in List-Language Override	If you enable language override, then you can set this property to one of the following values: ◦ **FALSE.** Hides the column in the Siebel client. ◦ **TRUE.** Displays the column in the Siebel client. For more information, see *Setting the Language Mode*.

Fixing Orphaned String References After an Upgrade

An upgrade from one release of Siebel CRM to another release can result in some string references disappearing. The Fix Strings Utility allows you to locate these orphaned strings and update them with new references. This utility uses the Siebel Tools Fix String References business service and the FixStringReferences business service method.

To fix orphaned string references after an upgrade

1. Navigate to the `$SIEBEL_HOME\BIN` folder.
2. Enter the following command:

```
consoleapp config_file app_lang user_Id password
"Siebel Tools Fix String References""FixStringReferences:Repository=repository_name,
PriorRepository=prior_repository_name, LogFile=log_file_name.log, FixReferences=true_or_false,
VerboseOutput=true_or_false, Object=object_type"
```

The following table describes the parameter values.

Parameter	Description
config_file	Specify the path to and name of the Siebel configuration file.
app_lang	Specify the current language, such as ENU.
user_Id	Specify the user Id for the repository that this utility searches.
password	Specify the password for the repository that this utility searches.
repository_name	Required. Specify the name of the repository that includes the string references to fix.
PriorRepository	Specify the name of the repository that your deployment used before the upgrade. If you set the FixReferences parameter to TRUE, then you must include the PriorRepository parameter.
log_file_name	Required. Specify the name of the log file. This utility writes this log file to the current working folder. You can enter an explicit path for this log file.
FixReferences	You can set this parameter to one of the following values: ○ **TRUE.** Fix invalid references. ○ **FALSE.** Save invalid references to the log file. This is the default value.
VerboseOutput	You can set this parameter to one of the following values: ○ **TRUE.** Write progress information to the command line. ○ **FALSE.** Do not write progress information to the command line. This is the default value.
object_type	Specify the object type, such as Applet. The utility finds invalid string references that this object type references. If you do not include this parameter, then the utility finds invalid string references for all object types.

Example Commands That Fix Orphaned String References After an Upgrade

The following example runs the utility for the Business Service object type, writes progress output to the command line, and writes information to the `fixstrings.log` file:

```
consoleapp $SIEBEL_HOME\bin\enu\tools.cfg ENU jgolding db2
"Siebel Tools Fix String References" "FixStringReferences:Repository=Siebel Repository,
LogFile=fixstrings.log, FixReferences=false,VerboseOutput=true, Object=Business Service"
```

The following example runs the utility for all object types and writes the results to the `fixstrings.log` file:

```
consoleapp $SIEBEL_HOME\bin\enu\tools.cfg ENU jgolding db2
"Siebel Tools Fix String References" "FixStringReferences:Repository=Siebel Repository,
LogFile=d:\temp\fixstrings.log, FixReferences=false"
```

Using the Locale Management Utility

This topic describes how to use the Locale Management Utility. It includes the following information:

- *Finding Untranslated Text Strings*
- *Finding Existing Translations*
- *Finding Objects That the Locale Management Utility Modifies*
- *Finding Objects Modified Since the Last Export*
- *Exporting Text Strings and Locale Properties to a File*
- *Importing Text Strings and Locale Properties*
- *Modifying the Value in All Strings for a Language*
- *Using the Command Line to Run the Locale Management Utility*

The *Locale Management Utility* (LMU) is a utility that you can use to manage how you configure Siebel CRM to localize text strings (such as field labels) and other locale properties (such as the height and width of controls). You can use it to export text strings to a file. After Siebel modifies or translates these strings in the file, you can use the Locale Management Utility to import them back into the repository. The Locale Management Utility also includes search and comparison tools.

Finding Untranslated Text Strings

You can use the Locale Management Utility to find text strings in the repository that Siebel has not translated or to find text strings that must be retranslated because Siebel modified the source string since the last translation. The Locale Management Utility searches or compares objects. It does not search or compare properties. If a locale object includes multiple string properties, then the search returns all strings that the locale object contains, even if only one of them is translated.

To find untranslated strings (Siebel Tools)

1. Click the Tools menu, click Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
3. In the Objects section, choose the applications or projects that you want to localize.
4. Click the Untranslated Strings tab.

   The Untranslated Strings tab includes an Attribute column. Siebel CRM uses the terms attribute and property. These terms have the same meaning.
5. (Optional) To display strings that are marked as Redo, make sure the following check box contains a check mark:

   Report String Attributes of Objects Marked With 'Redo' Fag

For more information, see *How the Locale Management Utility Indicates That Siebel CRM Modified a Record Since the Last Export*.

6. Click Find Strings.

   The Locale Management Utility searches the properties of the string objects. It searches the objects only in the application or project you choose. It displays object definitions that include strings that are not translated. If you choose to display Redo objects in Step 6, then it also displays the strings that you must retranslate.
7. (Optional) To do more, you can click the following buttons:
   - **Find Views.** Find the views that reference the untranslated strings.
   - **Go To.** You choose an object in the Results list, and then click Go To. The Object Explorer displays the object definition that includes the string.
   - **Export.** Export all untranslated strings to a file.
8. (Optional) Determine if an existing translation exists that you can use for the untranslated strings that the Locale Management Utility shows in Step 7.

   For more information, see *Finding Existing Translations*.

Finding Existing Translations

To find an existing translation that you can use for an untranslated string, you can use the Locale Management Utility to search the objects in the repository. You can reuse an existing translation for an object that you create or modify.

To find existing translations (Siebel Tools)

1. Click the Tools menu, click Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
3. In the Objects section, choose the application or project that you want to localize.
4. Click the Untranslated Strings tab.
5. Click the Find Translations button.

   After a few moments, the Locale Management Utility displays the results. For more information, see *How the Locale Management Utility Finds Existing Translations*.

How the Locale Management Utility Finds Existing Translations

The Locale Management Utility compares an untranslated string to the properties that might contain this string in other objects in the repository. If it finds an object that includes the same string, then it searches for a translation in the language that you choose as the target language. If it finds a translation, then it displays the best candidates for translation in the Results list and allows you to export it to a file.

For example, you choose English-American as the source language and Spanish as the target language. You create an applet with a title of Customer that is not translated. If you click Find Translation in the Locale Management dialog box, then the Locale Management Utility searches the repository for other objects that include a property that includes the Customer text string. If it finds a match, then it searches for a Spanish translation of this string. If a translation already exists, then it displays this translation and you can export it to a file.

How the Locale Management Utility Chooses Among Multiple Translations

If the Locale Management Utility finds more than one translation for a source string, then it does the following:

- If the source string resides in the property of an object that is related to a business component, such as Control Caption or List Column Display Name, then the Locale Management Utility examines translations from the same business component first, and then does the following:
  - If multiple translations exist in the same business component, then it chooses the string that occurs most frequently.
  - If no translations exist in the same business component, then it chooses the translation that occurs most frequently in all business components.

    For example, Applet A references the Account business component. Applet A includes a control caption with the value of Account, and this value is translated to Account_FRA for French. You create a new applet, Applet B, that also references the Account business component, and Applet B also includes a control caption with the value of Account. If you run Find Translations, then the Locale Management Utility finds Account_FRA as an existing translation and chooses it as the best candidate for this string.
- If the source string is not a property related to a business component, such as Menu Item Caption, then the Locale Management Utility chooses the translation that occurs most frequently.

Finding Objects That the Locale Management Utility Modifies

This topic describes how to find objects that the Locale Management Utility modifies.

To find objects that the Locale Management Utility modifies (Siebel Tools)

1. Click the Tools menu, click Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
3. In the Objects section, choose the application or project that you want to localize.
4. Click the Modified Objects tab.
5. In the Search Criteria section, make sure the Changed Since check box includes a check mark.
6. Click the Changed Since drop-down arrow, and then choose the date that the Locale Management Utility must use as the starting date for the search.
7. Click Start.

Finding Objects Modified Since the Last Export

You can use the Locale Management Utility to find objects in the repository that were modified since the last time you exported strings. This search might be useful if your development and localization efforts occur simultaneously. It helps you keep strings in the repository synchronized with the strings that you export to a file for localization.

To find objects modified since the last export (Siebel Tools)

1. Click the Tools menu, click Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
3. Click the Modified Objects tab.

4. (Optional) In the Search Criteria section, set the Locale Management Utility to search from a date:
   - Make sure the Changed Since check box includes a check mark.
   - Click the Changed Since drop-down arrow, and then choose the date that the Locale Management Utility must use as the starting date for the search.

     If you use the Changed Since option, and if you click Start, then the Locale Management Utility marks all records it returns for a modified project as Redo regardless of whether Siebel CRM modified a locale property. It does this because it searches for modifications that reside at the object level, not at the property level. For more information, see *How the Locale Management Utility Indicates That Siebel CRM Modified a Record Since the Last Export*.
5. (Optional) In the Search Criteria section, set the Locale Management Utility to compare objects in the repository to objects in a source file:
   - Make sure the Different From File check box includes a check mark.
   - Click Browse, and then choose the source file.
6. Click one of the following buttons:
   - **Start.** Finds records that match the search criteria, flags the records that it returns as Redo, and then displays the results.
   - **Preview.** Finds records that match the search criteria and displays the results. It does not flag records as Redo.
7. (Optional) To do more, you can click the following buttons:
   - **Save.** Saves a result set in a log file.
   - **Go To.** The Object Explorer displays the parent object of the string or property.
   - **Load.** Imports a result set from a previously saved file. After the Locale Management Utility displays this result set in the Results list, you can use Go To to examine each record.

How the Locale Management Utility Indicates That Siebel CRM Modified a Record Since the Last Export

If Siebel CRM modified a record in the repository since the last export, then it might require another translation. The Locale Management Utility uses the Redo flag to mark this record.

If you use the Locale Management Utility to import records, then it compares the source language records in the repository with the source language records in the import file. If Siebel CRM modified the records in the repository since the export occurred, then this utility uses the Redo flag to mark the target language records. This configuration helps you identify records that might require another translation. For more information, see *Importing Text Strings and Locale Properties*.

Exporting Text Strings and Locale Properties to a File

You can use the Locale Management Utility to export strings and other locale properties to a file. This file can use any one of the following file types:

- `.slf`
- `.txt`
- `.xlf`

You cannot use the Locale Management Utility to export to a Microsoft Excel `.xls` file.

To export text strings and locale properties to a file (Siebel Tools)

1. Click the Tools menu, click Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, select the source language and the target language.

   To export text strings and locale properties, the language mode and the Locale Management Utility source language must use the same language. For more information, see *Setting the Language Mode*.
3. In the Objects section, choose the application or project that the Locale Management Utility must export.
4. Click the Export Tab.
5. Click String Attributes Only or All Localizable Attributes.

   A localizable property can include translatable strings and other locale properties, such as the width and height of controls. Each locale property might contain a different value that meets the need for a specific locale. Siebel CRM uses the terms attribute and property. These terms have the same meaning.
6. Click Export.
7. In the Save As dialog box, choose the folder where the Locale Management Utility must export the file.

   For example, choose `$SIEBEL_HOME\OBJECTS\language_code` where *language_code* is the target language you selected in Step 2.
8. Enter a file name, choose a file type, and then click Save.
   - If you click String Attributes Only in Step 5, then the available file types are `.txt` or `.xlf`.
   - If you click All Localizable Attributes in Step 5, then the available file type is `.slf`.

Importing Text Strings and Locale Properties

You can use the Locale Management Utility to import translated strings and other locale properties into the repository. You can preview the import before the utility actually does the import.

To import text strings and locale properties (Siebel Tools)

1. Click the Tools menu, click Utilities, and then click Locale Management.
2. In the Locale Management dialog box, in the Options tab, in the Languages section, choose the source language and the target language.
3. Click the Import tab.
4. Enter the name of the file that the utility must import.

   Use the Browse button to choose the file. The default file name is one of the following:
   - `Results.txt` if the file includes strings only
   - `Results.slf` if the file includes all locale properties

   If you import an XML Localization Interchange Field file (`.xlf`), then make sure a working Internet connection exists during the import.
5. (Optional) To preview the import, do the following:
   a. Enter the path and name of the file where the utility must store the results for previewing.

The default file name is `preview.log`.

b. Click Preview.

   The Locale Management Utility writes the results of the import to the log file. It does not write the results to the repository. It does not mark modified records with a Redo flag during preview.

6. (Optional) To mark records that Siebel CRM modified since the export occurred, make sure the following check box contains a check mark:

   Mark Changed Records with Redo Flag

   For more information, see *How the Locale Management Utility Indicates That Siebel CRM Modified a Record Since the Last Export*.
7. Click Import.

   The utility imports the locale properties into the repository. It creates the following log file in the `$SIEBEL_HOME\OBJECTS` folder:

   ```
   LMUImportTruncation.log
   ```

   This file includes detailed information and error messages about records that the utility did not import.

Modifying the Value in All Strings for a Language

You can use the Locale Management Utility to replace strings in bulk. For example, you want to configure Siebel CRM to modify all text strings that include Accounts to Companies for the English locale. You can use the Locale Management Utility to export the strings to a file, modify the file so that it includes only Companies instead of Accounts, and then import these strings into the repository.

Using the Locale Management Utility to replace strings is most useful for strings that Siebel CRM stores in string-override fields. For information about modifying symbolic strings, see *Modifying a Symbolic String to Globally Update Display Values*.

To modify the value in all strings for a language

1. Export the strings you want to modify to a file.

   The source language and the target language cannot be the same language. For more information, see *Exporting Text Strings and Locale Properties to a File*.
2. Use a text editor to modify the strings in the file that you created in Step 1:

   - Remove strings that you do not want to replace from the file.
   - In the Target String column of the file, enter the new string that replaces the old value.
3. Import the file.

   For more information, see *Importing Text Strings and Locale Properties*.

Using the Command Line to Run the Locale Management Utility

You can use the command line to run the Locale Management Utility. The commands that this topic describes use the following format:

- `xxx`. A placeholder for a required parameter.
- `[xxx]`. A placeholder for an optional parameter.
- `xxx|yyy`. An OR condition (for example, `xxx` or `yyy`).

You must include the absolute path with any file name that you specify. For example, if you specify the export file as `results.txt`, then the utility creates this file in the current folder. In this example, if the installation folder is $SIEBEL_HOME, then the utility creates this file in the following folder:

```
$SIEBEL_HOME\BIN
```

It does not create it in the following folder:

```
$SIEBEL_HOME\OBJECTS
```

Using the Command Line to Export Strings and Locale Properties

The command that this topic describes allows you to export localizable properties for all projects or for all applications. You can specify one of the following values:

- **all.** Export all translatable properties and language override properties to a file that uses an .slf extension.
- **string.** Export only string properties to a file that uses a .txt or .xlf extension. If you do not specify a file name, then the utility displays an error.

Format:

This command uses the following format:

```
/lmu source_language target_language export proj|app all|string file [project_file]
```

Project Parameter:

The Locale Management Utility includes the following parameter:

```
project_file
```

You can use it to identify the projects to export. This parameter identifies the name of an ASCII text file that includes a list of projects that are delimited by a line feed. If you do not include the project_file parameter, then the utility exports all projects.

You can use the `proj|app` parameter to identify the projects or applications that include the strings to export. If you use the project_file parameter, then you must choose `proj`. If you choose `app`, and if you include the project_file parameter, then the utility ignores this file.

Example:

The following command exports properties:

```
Siebdev /c tools.cfg /u sadmin /p ******** /d serverdatasrc /lmu ENU DEU export proj
```

```
string C:/lmu/lmu_1.xlf /ws "ws1" "Siebel Repository"
```

This example exports all string properties and language override properties for the projects that the following file lists:

```
C:temp\proj_to_exp.txt
```

It exports these properties to the following file:

```
C:\temp named my_proj_results.txt
```

The source language of the file is English-American (ENU) and the target language is French (FRA).

Using the Command Line to Import a File

The command that this topic describes allows you to import a file. If the source string in the import file is different than the source string in the repository, then it marks the target locale object as Redo.

Format:

This command uses the following format:

```
/lmu source_language target_language import file
```

Example:

The following command imports a file:

```
siebdev /u sadmin /p ******** /d server /lmu ENU FRA import "$SIEBEL_HOME\objects\results.slf"
```

This example imports the `results.slf` file from the following folder. This folder is the installation folder for an earlier version of Siebel:

```
$SIEBEL_HOME\objects
```

The source language of the file is English-American (ENU) and the target language is French (FRA).

The file includes all localizable string properties and language override properties.

Using the Command Line to Export Strings That Require Translation

The command that this topic describes allows you to export all untranslated strings and strings marked with the Redo flag to a file. It can export all projects or all applications. The exported file includes the related view names.

Format:

This command uses the following format:

```
/lmu source_language target_language todo proj|app [file]
```

Example:

The following command exports strings that require translation:

```
Siebdev /c tools.cfg /u sadmin /p ******** /d serverdatasrc /lmu ENU DEU export proj
string C:/lmu/lmu_1.xlf /ws "ws1" "Siebel Repository"
```

This example finds all untranslated strings and redo strings for all applications. It exports the results to the following file:

```
$SIEBEL_HOME\objects\results.txt
```

The source language is English-American (ENU) and the target language is French (FRA).

11 Testing and Troubleshooting Your Customizations

Testing and Troubleshooting Your Customizations

This chapter describes how to test and troubleshoot your customizations. It includes the following topics:

- *About Testing and Troubleshooting Your Modifications*
- *Setting Debug Options to Open the Siebel Client*
- *Recovering from a Failed Development Server*

About Testing and Troubleshooting Your Modifications

You can use a local instance of the Siebel Web Client that runs on your computer to test your modifications. For more information about installing the Siebel Web Client, see the *Siebel Installation Guide* .

Before testing your changes, set the debug options as shown in *Setting Debug Options to Open the Siebel Client*. For more information about using debug windows, see the following topics:

- *Using the Call Stack Window*
- *Using the Watch Window*

Setting Debug Options to Open the Siebel Client

The debug options allow you to modify the run time settings that open an instance of the Siebel Web Client in the following situations:

- You enable Auto-start Web Client.
- You click the Debug menu and then click Start to start an instance of the Siebel Web Client. You typically do this if you want to debug Siebel eScript or Siebel VB. For more information, see *Siebel eScript Language Reference* or *Siebel VB Language Reference* .

To set debug options to open the Siebel client (Siebel Tools)

1. Click the View menu and then click Options.
2. In the Development Tools Options dialog box, click the Debug tab.
3. Set the debug options as required.

   Make sure you set the following options:

   - Executable

- CFG File
- Browser
- Working Directory

For more information, see *Development Options for Debugging*.

Recovering from a Failed Development Server

This topic describes how to recover from a failed development server or failed repository when you cannot access a backup copy of this server or repository. It requires that a recent image of the repository exist on the computer that a remote user uses, and that this user performed a full Get to create this repository.

To recover from a failed development server

1. Log in to Web Tools on the remote computer that includes the intact repository.
2. Archive the projects that this repository contains.

   For more information, see *Exporting Objects to an Archive File*.
3. Create a new repository:

   - To create a repository record, add a new repository object.
   - To create the schema in the database for the repository, run `imprep.ksh` for Linux or UNIX.

     For more information, see *Exporting and Importing Repositories*.
4. Import the archive you created in Step 2 into the repository you created in Step 3.

   For more information, see *Importing Objects From an Archive File*.
5. Test the repository you created in Step 3 and make sure it includes the expected functionality.
6. Check projects into the Siebel Server.

12 Archiving Objects

Archiving Objects

This chapter describes how to archive objects. It includes the following topics:

Note: Archiving is not supported nor intended for the migration of entire repositories from one environment to another. For more information, see *Exporting and Importing Repositories*.

Overview of Archiving Objects

Archiving is a process where you export objects from the repository to an archive file. You can then import objects from the archive file back into the repository. You can use an archive to back up sets of objects or move them to another environment that shares the same physical database schema as the source environment. Note the following:

- An archive does not depend on a database because it includes only repository information. You can use it to exchange repository data between environments with different database platforms, including local and server databases, as long as these databases use the same schema.
- If you import objects from an archive, then you can specify conflict resolution rules for each object. You can specify to ignore an imported object, replace an existing object with an imported object, or merge two objects according to object properties.
- You can archive individual objects or entire projects to an archive.
- You can use code control software to control an archive.
- If you want to back up or move the entire repository to another environment, then see *Exporting and Importing Repositories*.

  Note: Archiving is not supported nor intended for the migration of entire repositories from one environment to another.

For more information about the files that you use when archiving objects, see *Files That You Use to Manage Repositories*.

Using Export and Import for Different Versions of Siebel CRM

If you want export objects from one version of Siebel CRM to another version, then do not import those objects through .sif files into the different version of Siebel CRM because Siebel might modify object definitions between these versions. If you import an object that is not valid, then an invalid configuration might result. Oracle does not support an invalid configuration.

Exporting Objects to an Archive File

This topic describes how to export objects to an archive (SIF) file. It includes the following information:

- *Exporting Objects to an Archive*
- *Using the Command Line to Export Objects to an Archive*
- *Exporting Workspaces to an Archive*
- *Using the Command Line to Export Workspaces to an Archive*
- *Input File That Batch Export Uses*

Note: Archiving is not supported nor intended for the migration of entire repositories from one environment to another. For more information, see *Exporting and Importing Repositories*.

Exporting Objects to an Archive

You can export top-level objects to an archive (SIF file), including business components, applets, views, and projects. You can also export various objects from different Workspaces to an archive file. Child objects are exported and imported with their parents.

The following procedure shows how to export objects, across different Workspaces, to an archive in Web Tools (the procedure is similar in Siebel Tools). This task is a step in *Roadmap for Setting Up and Using Siebel Tools*.

To export objects to an archive

1. In the Object Explorer and Object List Editor, locate and select the objects you want to export.

2. In the Archive menu, select Add To Archive (Alt+N is the keyboard shortcut) .

   In the Objects to Archive window that opens, the objects that you selected in step 1 are added to the Objects to Archive list.

   The Objects to Archive list stores the following information about each object:

   - **Name** of object, for example: SIS Account List Applet.
   - **Type** of object, for example: Applet.
   - **Workspace Name**, for example: dev_sadmin_00.
   - **Workspace Version**, for example: 1.

   The Objects to Archive window has the following buttons:

   - **Collapse**. Click to hide the Objects to Archive window; click again to show the window.
   - **Close** (the X icon). Click to close the Objects to Archive window altogether. The information in the window will remain the same until the window opens again.
   - **Delete** (the trash can icon). Click to remove a selected item from the Objects to Archive list and then click OK on the confirmation prompt that appears.
   - **Export** (the download icon). Click to export the items in the Objects to Archive list to a SIF file.
   - **Clear** (the double trash can icon). Click to remove all items from the Objects to Archive list.
3. Repeat steps 1 and 2 as required to add further objects to the archive.
4. To add an object from a different Workspace to the archive:

   - Click Workspace (the cube icon) to go to the *Workspace Dashboard*.
   - In the Workspace Explorer, select the Workspace that you want and then click Open.
   - Close the *Workspace Dashboard* and then repeat steps 1 and 2.
5. Click Export (the download icon) to export the items in the Objects to Archive list to a SIF file:

   - When prompted, enter a file name for your SIF file. For example: objects.sif.
   - Click Save to save the archive file.

Using the Command Line to Export Objects to an Archive

The following procedure shows how to export objects to an archive (SIF file) using siebdev from the command line.

To export objects to an archive using the siebdev command

1. Open a command line and then navigate to the `$SIEBEL_HOME\BIN` folder.
2. Use the following command to run the siebdev.exe file:

```
siebdev /c config_file /d database /u user_name /p password /ws Workspace_name
```

```
/batchexport repository_name input_file_name log_file
```

The following example exports the objects that the obj.txt input file specifies. It logs results to the export.log file:

```
siebdev /c tools.cfg /d sample /u sadmin /p ******** /ws MAIN
/batchexport "Siebel Repository" obj.txt export.log
```

For more information about the input file, see *Input File That Batch Export Uses*.

Exporting Workspaces to an Archive

You can export entire Workspaces to an archive (SIF file). You can also combine adding objects and entire Workspaces to an archive. The following procedure shows how to export objects and Workspaces to an archive in Web Tools (the procedure is similar in Siebel Tools).

To export Workspaces to an archive

1. In the Object Explorer and Object List Editor, locate and select the objects you want to export.
2. In the Archive menu, select Add To Archive (Alt+N is the keyboard shortcut).

   In the Objects to Archive window that opens, the objects that you selected in step 1 are added to the Objects to Archive list.

   The Objects to Archive list stores the following information about each object:

   - **Name** of object, for example: SIS Account List Applet.
   - **Type** of object, for example: Applet.
   - **Workspace Name**, for example: dev_sadmin_00.
   - **Workspace Version**, for example: 1.

   The Objects to Archive window has the following buttons:

   - **Collapse**. Click to hide the Objects to Archive window; click again to show the window.
   - **Close** (the X icon). Click to close the Objects to Archive window altogether. The information in the window will remain the same until the window opens again.
   - **Delete** (the trash can icon). Click to remove a selected item from the Objects to Archive list and then click OK on the confirmation prompt that appears.
   - **Export** (the download icon). Click to export the items in the Objects to Archive list to a SIF file.
   - **Clear** (the double trash can icon). Click to remove all items from the Objects to Archive list.
3. Repeat steps 1 and 2 as required to add further objects to the archive.
4. To add the entire contents of a Workspace to the archive:

   - Click Workspace (the cube icon) to go to the *Workspace Dashboard*.
   - In the Workspace Explorer, locate and open the Workspace that you want.
   - In the Archive menu, select Export Workspace to Archive.

     The entire Workspace and all its objects are added to the Objects to Archive list.

Repeat this step as required to add multiple Workspaces to the archive. If an object in a Workspace has already been added to the archive, then it will not be added again.

5. Click Export (the download icon) to export the items in the Objects to Archive list to a SIF file:
    - Enter a file name for your SIF file. For example: objectscombined.sif.
    - Click Save to save the archive file.

Using the Command Line to Export Workspaces to an Archive

The following procedures show how to export whole Workspace objects or any particular object under a Workspace, which are Workspace-enabled, to an archive (SIF file) using siebdevcli from the command line. Note that siebdevcli is supported on both the server (Windows and UNIX platforms) and client.

To export whole Workspaces to an archive using the siebdevcli command

1. Open a command line and then navigate to the `SIEBEL_HOME\BIN` folder.
2. Use the following command to run the siebdevcli.exe file:

```
siebdevcli /c config_file /l language_code /u user_name /p password /d odbc_datasource
/WorkspaceExport /Repository "Siebel Repository" /Workspacename Workspace_name
/f object_export_file /ToVersion 1111
```

The following example exports the modified objects that belong to MAIN and writes the file provided in the `/ExportFile` parameter.

```
siebdevcli /c tools.cfg /l ENU /u sadmin /p ******** /d oradev109
/WorkspaceExport /Repository "Siebel Repository" /Workspacename "MAIN"
/f wsexport.sif /ToVersion 9
```

The following table describes some of the parameters that you can use with the siebdevcli command line.

Parameter	Optional	Description
`/WorkspaceExport`	Yes	Exports a whole Workspace. This parameter is only required when running siebdevcli in Workspace Export Mode (that is, when doing a Workspace export). This option is not required if running Import Repository Mode.
`/Repository` or `/y`	No	The name of the repository.
`/WorkspaceName`	No	The name of the Workspace.
`/ExportFile` or `/f`	No	The name of the SIF file.
`/ToVersion`	Yes	The end version. The default is 99999.

Parameter	Optional	Description

To export objects under a Workspace to an archive using the siebdevcli command

1. Open a command line and then navigate to the `SIEBEL_HOME\BIN` folder.
2. Use the following command to run the siebdevcli.exe file:

```
siebdevcli /c config_file /l language_code /u user_name /p password /d odbc_datasource
/WSObjectExport /Repository "Siebel Repository" /Workspacename Workspace_name
/i object_input_file /l log_file_name
```

The following example exports the objects that the `infile.txt` input file specifies. It logs results to the `export.log` file:

```
siebdevcli /c wsexport.cfg /l ENU /u SADMIN /p ******** /d oradev109
/WSObjectExport /y "Siebel Repository" /Workspacename "dev_sadmin_ws"
/i infile.txt /l export.log
```

The following table describes some of the parameters that you can use with the siebdevcli command line.

Parameter	Optional	Description
`/WorkspaceExport`	No	Exports whole objects. This parameter is only required when running siebdevcli in Workspace Export Mode (that is, when doing a Workspace export). This option is not required if running Import Repository Mode at the same time.
`/Repository` or `/y`	No	The name of the repository.
`/WorkspaceName`	No	The name of the Workspace.
`/InputFile` or `/i`	No	The name of the input file.
`/LogFile` or `/l`	No	The name of the log file.

Input File That Batch Export Uses

The batch export switch uses an input file that specifies the objects to export. This input file uses the following format. You cannot use a space before or after a comma in this file:

```
object_type,query_expression,file_name.sif
```

Where:

- `object_type` identifies the object type to export, such as Business Component.
- `query_expression` can include any query that Siebel supports. It can also contain just a simple object name, such as Account. For more information, see *Using the Query Menu to Run a Query*.
- `file_name.sif` can use an absolute file path or a relative file path to the current folder.

For example:

```
// The following exports the Account BusComp:
Business Component,Account,export.sif

// The following exports all BusComps containing the string "Contact":
Business Component,*Contact*,export.sif

// The following exports all BusComps modified since October 1st:
Business Component,[Updated] > '10/1/2021',export.sif
```

You can include multiple lines in the input file and each of these lines can specify to export multiple objects to a different SIF file. If you specify to export to the same SIF file in multiple lines, then the last line is the only one that will actually make it, but it *will* run all lines. For example, the following line from an input file will create a file named "export.sif" with the Account BusComp in it. It will then overwrite it with a file that has the Contact BusComp in it.

```
Business Component,Account,export.sif
Business Component,Contact,export.sif
```

Importing Objects From an Archive File

This topic describes how to import objects from an archive (SIF) file. It includes the following information:

- *Preparing Your Environment to Import Objects From Archive*
- *Locking Projects Before You Import an Archive*
- *Importing Objects From an Archive*
- *Options to Resolve a Conflict*
- *Import Wizard - Review Conflicts and Actions Page*
- *Using the Command Line to Import an Archive*

Note: Archiving is not supported nor intended for the migration of entire repositories from one environment to another. For more information, see *Exporting and Importing Repositories*.

Preparing Your Environment to Import Objects From Archive

You must prepare your environment before you import an archive.

To prepare your environment to import objects from an archive

1. Make sure the local computer can access the import file through a network or local drive.
2. Make sure the same schema version is being used by the repository you use to create the archive and the repository to which Siebel imports the archive.

You can export or import an archive only among repositories that use the same schema version. To determine the schema version that a repository uses, see *Viewing Information About the Current Repository*.

3. Make sure the repository is open in Siebel and that is is the active repository.

   For more information, see *Viewing Information About the Current Repository*.
4. Make sure the projects that the import affects are checked out.

   For more information, see *Locking Projects Before You Import an Archive*.

Locking Projects Before You Import an Archive

For non-Workspace enabled objects, you need to lock the project before importing an archive file. The Import wizard informs you of any unlocked projects that you must lock. It does this after it analyzes the objects in the archive and compares them to the objects that reside in the repository. For Workspace-enabled objects, object locking is done at the database level. Therefore, there is no need to lock any project or object for importing.

Importing Objects From an Archive

The *Import Wizard* is a task tool that you use to step through the process of importing objects from an archive file. The wizard has the following stages: Start, Review, Confirm, and Summary. To navigate forward and back through the wizard, use the Next and Previous buttons. Click Cancel at any time to cancel the import process. You can only import one archive file at a time using the Import Wizard.

The following procedure shows how to import objects from an archive file, containing various objects and/or Workspaces, into the repository in Web Tools (the procedure is similar in Siebel Tools).

To import an archive

1. Prepare your environment to import an archive – for more information, see *Preparing Your Environment to Import Objects From Archive*.
2. In the Workspace Explorer, create a new Workspace (for example: dev_sadmin_import).
3. In the Archive menu, select Import from Archive.
4. In the Import from Archive dialog that opens, click Choose File, browse to the archive file that you want to import (for example: objectscombined.sif) and then click Import.

   The Import Wizard - Preview (or *Start*) page opens with the following information:

   - **Objects in Archive**, which lists the projects and other top-level objects in the archive file.
   - **Conflict Resolution**, which has the following options: Overwrite, Merge, and Do Not Import.
5. To remove an object from the Objects in Archive list, select it and then click Delete (the trash can icon).

6. In the Conflict Resolution section, select one of the following options to specify what to do if both the archive file and the repository contain the same top-level object:
   - **Overwrite.** Select this option if you want to overwrite the objects in the repository with the objects in the archive file.
   - **Merge.** Select this option (which is the default) if you want to merge the object definitions in the archive file with the existing definitions in the repository.
   - **Do Not Import.** Select this option if you do not want to import the object definitions from the archive file, and continue to step 9.

   For more information about these options, see *Options to Resolve a Conflict*.
7. Click Next.

   The *Import Wizard - Review Conflicts and Actions Page* page opens. The purpose of this page is to highlight any *Object Differences* or *Attribute Differences* that exist between objects in the archive file and the repository. Review the information on this page and modify it, if required, as follows:

   a. Select an object in the Conflicting Objects Tree applet.
   b. In the Object Differences section, review the information for the selected object and, if required, modify the value in the Action field. To modify the value in the Action field, click the drop-down arrow in the field and select another value. For a description of these values, see *Options to Resolve a Conflict*.
   c. In the Attribute Differences section, review the information for the selected object (from the Object Differences section) and, if required, modify the value in the Resolution field. To modify the value in the Resolution file, click the drop-down arrow in the field and select another value. Note the following:

      - If the Action field contains the `Merge` value, then you can modify the resolution value. Click the drop-down arrow in the Action field and select either Repository or File.
      - If the Action field contains the `Repository` or `File` value, then you cannot modify the resolution value.
   d. Repeat step a, b, and c to review and, if required, modify the (conflicts and actions) information for each object in the archive file.
8. Click Next.

   The Import Wizard - Confirmation page opens, which:

   - Summarizes the operations that it is going to perform on the repository, for example, as follows:

     ```
     The operation is going to modify the repository as follows:
     2 objects will be inserted
     0 objects will be deleted
     1 attribute will be updated
     ```
   - Lists the objects that will be imported on clicking Next.
9. Click Next.

   The Import Wizard - Summary page opens. This page either displays the following message or summarizes why the import was unsuccessful:

   ```
   Successfully Imported.
   ```
10. Click Finish to exit the Import Wizard.

Options to Resolve a Conflict

A *conflict* is a situation that occurs if an object that you want to import from an archive does not match the corresponding object in the repository. The objects are examined to determine if they include the same properties, property values, and child objects. If these items are not the same, then this situation represents a conflict.

The following table describes the options available to resolve conflicts that occur when importing an archive file. It describes what happens if it is determined that both the archive file and the repository contain the same top-level object.

Option	Description
Overwrite – the object in the repository.	Selecting this option deletes the object (plus all child objects) in the repository and then copies the object (plus all child objects) from the archive file to the repository.
Merge – the object definitions from the archive with the definition in the repository.	Selecting this option does the following: • Replaces the properties that are different in the repository with the properties that the archive contains. • If the archive file includes child objects that the repository does not contain, then it copies these objects from the archive to the repository. • Does not modify child objects in the repository that are not also in the archive file. When the merge finishes, the top-level objects in the repository include the same properties and child objects that the object in the archive file includes. It also includes any child objects that already exist in the repository. Merge is typically the safest option. Objects are merged by default.
Do Not Import – the object definition from the archive	Selecting this option does not modify the objects in the repository.

Import Wizard - Review Conflicts and Actions Page

The Import Wizard - Review Conflicts and Actions page opens when a difference exists between an object in the archive file and the same object in the repository; and typically when you want to overwrite or merge objects during an import. This page contains the following elements – for information on how to review and (if required) modify the information on this page, see *Importing Objects From an Archive*.

- **Conflicting Objects Tree Applet.** A hierarchical list of objects (from all levels, not just top-level objects) that have conflicts. The hierarchy uses the same structure as the object type hierarchy in the repository but it displays only objects that have conflicts, rather than all repository objects.

- **Object Differences.** This section lists all objects that include a difference. The following table describes the fields in this section.

Field	Description
File	If this check box is selected, then the object exists in the archive file. An object can exist in both the archive file and in the repository.
Repository	If this check box is selected, then the object exists in the repository. An object can exist in both the archive file and in the repository.
Action	Indicates the proposed resolution for the object, which can be one of the following values: File, Repository, or Merge. Siebel uses the choice that you made in the Conflict Resolution section of the Import Wizard - Preview page (see step 5 in *Importing Objects From an Archive*) to determine what action to take (and specifies the action in this field). To modify the value in this field, click the drop-down arrow in this field and select another value. For a description of these values, see *Options to Resolve a Conflict*.

- **Attribute Differences.** This section displays the property value conflicts for the selected object in the Object Differences section. It lists only the properties that include a conflict. The terms attribute and property have the same meaning in Siebel CRM. The following table describes the fields in the Attribute Differences section.

Field	Description
Attribute	Name of the object.
File	Value of the object in the archive.
Repository	Value of the object in the repository.
Resolution	Specifies one of the following values: ◦ **File.** Deletes the value in the repository during the import and replaces it with the value from the archive file. ◦ **Repository.** Keeps the value (does not modify the value) in the repository during the import. Depending on the value in the Action field in the Object Differences section, you can change the value in the Resolution field as follows: ◦ If the Action field contains the `Merge` value, then you can modify the resolution value. Click the drop-down arrow in the Resolution field and then select either Repository or File. ◦ If the Action field contains the `Repository` or `File` value, then you cannot modify the resolution value.

Field	Description

Using the Command Line to Import an Archive

The following procedure shows how to import an archive (SIF file) using siebdev from the command line. You can also use the command line to do other work.

To import an archive using the siebdev command

1. If the archive file contains non-Workspace enabled objects, such as Table objects, then lock the projects for these objects. It is not possible to lock projects from the command line.
2. Use the following command to run siebdev.exe:

    ```
    siebdev.exe /c tools.cfg /d Datasource /u user /p password /ws Workspace
    /batchimport repository_name import_mode sif_files log_file
    ```

    Where:

    - `/ws` refers to the Workspace name and this Workspace needs to be in an editable state for import to be successful.
    - `import_mode` determines how to handle a conflict. You can use one of the following values with it: overwrite, merge, or skip. If you use `skip` and if a corresponding object does not exist in the repository, then there is no conflict and the utility imports the record. For more information, see *Options to Resolve a Conflict*.
    - `sif_files` specifies the name of one or more SIF files (for example: `sif_file1`, `sif_file2`, `sif_file_n`) that contains the objects that `batchimport` imports or the path to a folder that includes .sif files.
    - `log_file` specifies the name of the file that siebdev uses to log information.

      You can use the full path or the relative path to the current folder to specify the SIF file and the log file.

Examples of Using the Command Line to Import an Archive

The following example imports an archive. It uses the overwrite option for conflict resolution and it logs the results to the import.log file. For more information, see *Options to Resolve a Conflict*.

```
siebdev.exe /c tools.cfg /d ServerDataSrc /u SADMIN /p ******** /ws dev_sadmin_m6 /
batchimport "Siebel Repository" overwrite import1.sif,import2.sif" import.log
```

The following example imports all files to the `$TOOLS_HOME\objects` folder. It uses the merge option for conflict resolution and logs the results to the `import.log` file.

```
siebdev.exe /c tools.cfg /d ServerDataSrc /u SADMIN /p ******** /ws dev_sadmin_m4 /
batchimport "Siebel Repository" merge $TOOLS_HOME\objects import.log
```

To import an archive using the siebdevcli command

Importing an Archive can also be done using `siebdevcli` command line option.

Examples of using the command line to import an archive

The following example imports an archive. It uses the overwrite option for conflict resolution and it logs the results to the `import.log` file. For more information, see *Options to Resolve a Conflict*.

```
siebdevcli.exe /c tools.cfg /d ServerDataSrc /u SADMIN /p ******** /WS dev_sadmin_m6 /batchimport
"Siebel Repository" overwrite import1.sif,import2.sif" import.log
```

The following example imports all files from the `$TOOLS_HOME\objects` folder. It uses the merge option for conflict resolution and logs the results to the `import.log` file.

```
siebdev.exe /c tools.cfg /d ServerDataSrc /u SADMIN /p ******** /ws dev_sadmin_m4 /
batchimport "Siebel Repository" merge $TOOLS_HOME\objects import.log
```

Viewing and Resolving Conflicts after Siebel Archive File Imports in Web Tools

After the completion of RepositoryUpgrade or you have imported a Siebel Archive File (SIF) through the command line Interface (only siebdevcli command is supported), you can view the imported objects and their properties. To do this you use the SIF Attribute Differences report in Web Tools.

Note: This report only exists in Web Tools.

Using the SIF Attribute Differences Report

Here's how you can view attribute differences and resolve conflicts:

1. Open the editable Developer Workspace into which you imported the attribute differences.

   Note: If you have run RepositoryUpgrade, locate the parent Integration Workspace you used to run the RepositoryUpgrade utility. Under that integration workspace will be an Integration Workspace created by the RepositoryUpgrade utility named int_siebel_updateN (where N is a sequential number). Underneath int_siebel_updateN is the Developer Workspace to open.

2. Go to **Application Menu > Archive > SIF Attribute Differences** (Keep the developer workspace open).
3. When the report opens you will see a tree structure listing the changed object types such as Applet, Business Component, Workflow Process and so on.
4. Expand each object type to find the objects that have changed. Each object type can have multiple changed objects.
5. When you select a particular object, the Account Business Component for example, if any of the attributes at the top-level object have changed, you will see a list in the *Attribute Differences Applet* on the right.

6. For each attribute that has changed you will see the following:

   a. Attribute Name: Name of the attribute that has been changed such as Post Default Value for a Business Component Field.
   b. Pre-existing Value: Value that was in your Repository before you imported the SIF file or ran RepositoryUpgrade.
   c. Import Value: Value that has been imported from RepositoryUpgrade or a SIF import.
   d. Keep Pre-existing Value checkbox: Select this to keep the value that was in your Repository before the import. Leave it unchecked to allow the imported value to overwrite the existing value.
7. Expand each object type and its child object types and resolve each changed attribute by either selecting the check box to keep the Pre-existing Value or leaving it unchecked to allow the imported value to overwrite the existing value.
8. Your changes are automatically saved when you navigate to a different row, or you can manually save each record that you change.
9. If you need to resolve these conflicts in multiple sessions, you can return to this report later.
10. Once you have made all the changes for each attribute, click **Apply Changes** to save all your decisions in the current Developer Workspace.
11. Once you click *Apply Changes* the report closes, and you are returned to the View you were on in Web Tools.

    Note: You can still go back in and make multiple changes to the report. Nothing is final until you version the Developer Workspace. Once you have done that, all decisions in that Workspace are final.

12. Apply all the decisions that you have made to the Workspace using the **Apply Changes** button, you are ready to deliver the Workspace.
13. Deliver the Workspace to its parent `int_siebel_updateN` Workspace.
14. You are now ready to deliver `int_siebel_updateN` to its parent workspace.

Using the Application Deployment Manager

The *Application Deployment Manager* (ADM) is a feature that you can use to administer a Siebel CRM deployment. ADM is supported, for example, through the following business service: Siebel Tools Export Support for ADM.

ADM allows you to export individual objects to a hotfix or to export all objects that Siebel CRM modified after a particular date and time to a mid-level release. For more information, see *Siebel Application Deployment Manager Guide* .

13 Siebel Tools Reference Information

Siebel Tools Reference Information

The reference information in this chapter applies only to Siebel Tools. It includes the following topics:

- *Application-Level Menus*
- *Application Toolbars*
- *Dialog Boxes That Compare Objects*
- *Dialog Boxes That Set Development Options*
- *Parameters That Convert Symbolic Strings*

Application-Level Menus

This topic describes the application-level menus that are available in Siebel Tools.

- *File Menu*
- *Edit Menu*
- *View Menu*
- *Screens Menu*
- *Go Menu*
- *Query Menu*
- *Tools Menu*
- *Window Menu*
- *Workspace Menu*
- *Help Menu*

File Menu

The following table describes the menu items that are available in the File menu.

Menu Item	Description
Open Repository	If multiple repositories reside in the development database, then this menu item allows you to open a repository other than the repository that is currently open.
New Object	Starts the New Object Wizard that allows you to create a list applet, form applet, chart applet, tree applet, business component, report, table, command, picklist, MVG, or view.
Close (CTRL+F4)	Closes the Object List Editor.

Menu Item	Description
Save (CTRL+S)	Saves modifications that the current editing window contains if you edit in one of the following windows: Layout, Menu, or Basic Scripts.
Save All	Saves modifications in all open editing windows.
Import	Imports text from an external text file into the Siebel VB Editor window. This text must use an SBL file format. This format is created when text from the Siebel VB editor is exported.
Export	Allows you to create a text file in delimited or HTML format that lists the property values of an object or all objects that the Object List Editor currently shows.
Print Setup	Modifies the printer and printing options for printing diagrams from the object visualization view.
Print Preview	Opens a print preview window that displays an object visualization view.
Print (CTRL+P)	Prints the active object visualization view diagram.
Exit	Closes Siebel Tools.

Edit Menu

The following table describes the menu items that are available in the Edit menu. The Edit menu items apply to individual objects in the Object List Editor. You can right-click an object in the Object List Editor to display a list of menu items.

Menu Item	Description
Undo (CTRL+Z)	Reverses the last modification you made to a property value in the Object List Editor or the Property window before you save the object.
Redo (CTRL+Y)	Reapplies modifications after the Undo command runs.
Undo Delete	This menu item appears if you delete a record in the Object List Editor. It allows you to undo the deletion.
Undo Record	If you have not saved the record, then this menu item removes a new object that you create or reverses modifications you made to an existing object.
New Record (CTRL+N)	Creates a new object in the Object List Editor and positions the cursor in the first required property.

Menu Item	Description
Copy Record (CTRL+B)	Creates a new object that is a copy of the object that you choose. It also creates copies of all child objects. It is recommended that you use the Copy Record menu item only if reusing an existing object is not practical.
Delete Record (CTRL+D)	Deletes the object that you choose and the child objects of the object you choose. It is recommended that you do not use the Delete Record menu item. Instead, use the Inactive property. For more information, see *About the Inactive Property*.
Cut (CTRL+X)	If you use this menu item while your cursor is in a property that contains text, then the text you choose is copied to the clipboard and the existing text is deleted. In the Applet Designer, it copies the control you choose to the clipboard and deletes the existing control.
Copy (CTRL+C)	If you use this menu item while your cursor is in a property that contains text, then the text you choose is copied to the clipboard. In the Applet Designer, it copies the control you choose to the clipboard.
Paste (CTRL+V)	Inserts text from the clipboard into a property at the insertion point. Inserts a control from the clipboard in the Applet Designer.
Delete (DEL)	If you use this menu item while your cursor is in a property that contains text, then the text you choose is deleted. In the Applet Designer, it deletes the control you choose.
Select All (CTRL+A)	Chooses all items. In the Applet Designer, it chooses all controls that the applet contains.
Change Records	Modifies multiple records simultaneously.
Find (CTRL+F)	Finds the text that you specify in the Siebel Script Editor window.
Replace (CTRL+H)	Replaces the text that you specify with different text in the Siebel Script Editor window.

View Menu

The following table describes the menu items that are available in the View menu. You use these menu items to display windows, toolbars, or visualization views.

Menu Item	Subitem	Description
Windows	Palette	Displays the Palettes pane.
Windows	Properties Window	Displays the Properties pane.
Windows	Applets Window	Displays the Applets window.
Windows	Controls Window	Displays the Controls/Columns window.

Menu Item	Subitem	Description
Windows	Bookmarks Window	Displays the Bookmarks window.
Windows	Web Templates Window	Displays the Web Templates Explorer window.
Windows	Multi Value Properties Window	Displays the Multi Value Property Window pane.
Windows	Refresh Windows	Requeries and updates the state of dockable windows.
Windows	Reset Windows	Closes all dockable windows except the Object Explorer for the currently active editor. Does not close editor windows.
Editors	Web Applet Editor	Opens the applet you choose in the Applet Web Template Editor, including the Controls/Columns window and Palettes pane.
Editors	Server Script Editor	Opens the Siebel Script Editor. You can specify the editor to use or you can use the default.
Editors	Browser Script Editor	Opens the Siebel Web Script Editor that you use to access the scripts that control the presentation and behavior of applet controls and list columns in a Web applet template.
Visualize	View Details	For more information, see *Viewing Object Relationships*.
Visualize	View Relationships	For more information, see *Viewing Object Relationships*.
Visualize	View Descendents	For more information, see *Viewing Object Relationships*.
	View Web Hierarchy	For more information, see *Viewing Object Relationships*.
Debug Windows	Calls (CTRL+L)	Opens the Call Stack window. This window displays the call stack of the Siebel VB script or the Siebel eScript script that you are currently debugging.
Debug Windows	Watch (SHIFT+F9)	Opens the Watch window. This window displays the values of local variables for items you are currently debugging, such as Siebel VB script, Siebel eScript, or Siebel Workflow.
Debug Windows	Errors	Opens the Errors window. This window displays the run- time errors in the Siebel VB script or Siebel eScript script that you are currently debugging.
Preview	N/A	Displays a preview of a Web view layout. This preview approximates how Siebel CRM displays the container page, screen bar, and view bar.
ActiveX Methods	N/A	Allows you to view the methods for the current ActiveX control in the Applet Designer.

Menu Item	Subitem	Description
Toolbars	N/A	Displays the following toolbars: Edit, History, List, Debug, Web Controls, and Configuration Context.
Status Bar	N/A	Displays the Status bar in the window.
Object Explorer (CTRL+E)	N/A	Displays the Object Explorer.
Options	N/A	Opens the Development Tools Options dialog box. For more information, see *Development Options for Debugging*.

Screens Menu

The following table describes the menu items that are available in the Screens Menu. The Screens Menu is available only if you log in as a system administrator.

Menu Item	Subitem	Description
Application Upgrader	Application Upgrade Object List	Displays the Application Upgrades list, Object Differences list, or Attribute Differences list in the Object List Editor. In the Attribute Differences list, you can right-click and select an option to show critical conflicts, non-critical conflicts, and all changes for the item selected in the Object Differences list.
Application Upgrader	Application Upgrade Database Version	For internal Oracle use.
Application Upgrader	Application Upgrade Attribute List	Displays the Application Upgrades list and the Attribute Differences list in the Object List Editor. In the Attribute Differences list, you can right-click and select an option to show critical conflicts, non-critical conflicts, and all changes for the item selected in the Application Upgrades list.
System Administration	System Preferences	Displays system preferences in the Object List Editor. This information is similar to the information that Siebel CRM shows in the System Preferences view in the Administration - Application screen in the Siebel client.
System Administration	Analytics Strings	For internal Oracle use.
System Administration	List of Values	Displays the lists of values that the development database contains.

Go Menu

The following table describes the menu items that you are available in the Go menu. The Go menu allows you to navigate records in a list.

Menu Item	Description
Back	Displays the previous screen in a sequence.
Forward	Displays the next screen in a sequence.
Previous Record (CTRL+UP)	Navigates to the first object that is preceding the current selection.
Next Record (CTRL+DOWN)	Navigates to the first object that is succeeding the current selection.
First Record (CTRL+PAGE UP)	Navigates to the first object in the list.
Last Record (CTRL+PAGE DOWN)	Navigates to the last object in the list.
Add Bookmark	Displays the Add Bookmark dialog box that allows you to create a bookmark to the object you choose. You can use this menu item to create a *bookmark*, which is a feature that allows you to return directly to an item.
Bookmark List	Displays the Bookmarks dialog box that allows you to choose an existing bookmark. You can also use this dialog box to rename or delete an existing bookmark.

Query Menu

The following table describes the menu items that are available in the Query menu. This menu allows you to create and refine an Object List Editor query. You can use this query to filter the list of objects that appear in the current Object List Editor.

Menu Item	Description
New Query (CTRL+Q)	Allows you to filter the set of objects that appear in the Object List Editor.
Refine Query (CTRL+R)	Allows you to add more filters to the current query.
Execute Query (ENTER)	Runs the query.

Menu Item	Description
Sort Order	Opens the Sort Order dialog box. This dialog box allows you to specify the order that the Object List Editor uses to display the records.

Tools Menu

The following table describes the menu items that are available in the Tools menu.

Menu Item	Subitem	Description
Compile (F7)	N/A	Opens the Object Compiler dialog box to compile one or more projects to a repository.
Compile Selected Objects (CTRL+F7)	N/A	Opens the Object Compiler dialog box to compile the objects you choose to a repository.
Lock Project (ALT+L)	N/A	Locks the project that the object you choose references.
Unlock Project (ALT+U)	N/A	Unlocks the project that the object you choose references.
Add To Archive	N/A	Opens the Export To Archive dialog box to add the top-level objects that you choose or projects to an archive.
Import From Archive	N/A	Starts the Import wizard to import objects from an archive.
Compare Objects	Selected	Compares two objects that you choose. It uses a list of object names and properties to display similarities and differences.
Compare Objects	Selected vs. Repository	Compares the object you choose to the corresponding object in the repository and displays similarities and differences.
Compare Objects	Selected vs. Archive	Compares the object you choose to the corresponding object in an archive and displays similarities and differences.
Compare Objects	Archive vs. Archive	Compares two files that you choose and displays similarities and differences.
Convert to Grid Layout	N/A	Converts a form applet that uses a nongrid layout to grid layout.
Search Repository	N/A	Opens the Search Repository dialog box to search for objects according to the object name, another property, or the object type.

Menu Item	Subitem	Description
Validate Object	N/A	Validates the object you choose. Lists errors according to severity, rule number, object name, and error description. Allows you to modify options for rules, severity, and enforcement.
Upgrade	Maintenance Update	Not applicable starting with Siebel CRM version 8.0.
Upgrade	Prepare Repository	Used to upgrade from a Siebel CRM version that occurs before version 7.x to version 8.0. The Prepare Repository utility runs before it does a repository merge. It migrates strings from the S_MSG table, merges labels and fields, and merges templates to applets for the language that you specify. For more information, see *Siebel Database Upgrade Guide* .
Upgrade	Migrate ICL Objects to Standard	Applicable if you choose the Incorporate Custom Layout (ICL) option. You choose this option to preserve the layouts of custom objects during a previous upgrade. Before you can perform a subsequent upgrade, you must migrate the ICL objects to the predefined repository. For more information, see *About Predefined Objects* and *Siebel Database Upgrade Guide* .
Upgrade	Upgrade Application	Displays the Application Objects Upgrade List in the Application Upgrader screen and opens the Merge Repositories dialog box. You use this menu item to merge predefined and custom repositories. For more information, see *Siebel Database Upgrade Guide* .
Upgrade	Generate EIM Processing Columns	Opens the EIM Processing Column Generator dialog box. You can use this dialog box to create missing EIM processing columns and indexes after you merge the repository.
Upgrade	Web Client Migration	Used to upgrade from Siebel CRM version 6.x to version 7.x or version 8.0. It associates Web templates to a group of applets and views so that Siebel CRM can use them in the Siebel client. For more information, see *Siebel Database Upgrade Guide* .
Utilities	Generate Help IDs	Oracle uses this subitem internally to create the sshelp.hm file for *Siebel Tools Online Help*. This file includes information about context Id numbers and text help identifiers that the Help Id objects specify.
Utilities	Locale Management	Allows you to use the Local Management Utility to import or export translatable strings and locale properties.
Utilities	Map Fax Properties	If you click business component in the Object Explorer, then this menu item opens the Map Fax Properties dialog box for the business component that you choose. You can use this dialog box to create mappings between fields in the business component and sheet properties for the fax software. These mappings allow you to customize the fax cover sheet and the fax message.
Utilities	Export View Previews	Exports the view that the Preview mode of the View Layout Editor shows to an HTML file.

Menu Item	Subitem	Description
Utilities	Case Insensitivity	Opens the Case and Accent Insensitivity Wizard. It allows you to do a case-insensitive or accent-insensitive search on columns in the Siebel schema. For more information, see *Configuring Siebel Business Applications* and *Siebel Database Upgrade Guide* .
Utilities	Build Patch	Starts the Patch Builder wizard that allows you to create a patch file.
Utilities	Apply Patch	Opens the Apply Patch window that allows you to apply the patch.

Window Menu

The Window menu lists the currently open Object List Editor, Application Designer, visualization view, and other windows, and allows you to navigate to windows that Siebel Tools does not currently show. If one of these windows is open, then Siebel Tools displays the Close menu item as the first item. The Close menu item closes the window that is currently active.

Workspace Menu

The Workspace menu contains a number of options that are used to configure and manage Workspaces. The following table describes the available options.

Option	Description
Create	Select this option to create a new Workspace. For more information, see *Creating New Workspaces*.
Checkpoint	Select this option to check in the changes that you made to the current version of the Workspace. For more information, see *Performing the Checkpoint Version Process*.
Revert	Select this option to revert the changes that you made to the current version of that Workspace. For more information, see *Reverting to Previous Workspace Versions*.
Rebase	Select this option to apply and up-take the changes that were made in the parent Workspace into the current Workspace. For more information, see *Rebasing Workspaces*.
Merge Reports	Select this option to view and resolve the conflicts that occurred during the rebase process. For more information, see *Detecting Conflicts in Workspaces and Applying Resolutions*.
Submit for Delivery	Select this option to change the status of the Workspace and make it ready for delivering the changes to the MAIN Workspace. For more information, see *Submitting Workspaces for Delivery*.
Deliver	Select this option to deliver the changes in the Workspace to the MAIN Workspace. For more information, see *Delivering Workspaces*.

Option	Description
	Note: This option is available only to the user who is also the owner of the MAIN Workspace.
Compare	Select this option to view the differences that are made to the objects in two selected Workspace versions. For more information, see *Comparing Workspace Versions*.
Undo Submit for Delivery	Select this option to cancel the Workspace delivery process. For more information, see *Canceling the Workspace Delivery Process*.
Workspace Explorer	Select this option to display the Workspace Explorer pane.

Help Menu

Note: This information applies to Siebel Tools and Web Tools.

The following table describes the menu items available in the Help menu.

Menu Item (Siebel Tools)	Menu Item (Web Tools)	Description
Contents	Contents	Opens *Siebel Tools Online Help*.
Technical Support	Technical Support	Displays the Technical Support Information dialog box that includes information that Technical Support might require, such as the version number of your Siebel Tools/Web Tools installation.
About Record	About Record	Opens a dialog box that displays information about the current object, including the object creator and creation date.
About View	About View	Opens a dialog box that displays information about the current screen, business object, and view, including applet layout.
N/A	About Repository	Opens a dialog box that displays information about the Siebel Repository, Schema Version, Language, Runtime Version, and Published Type.
About Siebel	About Siebel	Opens a dialog box that displays Siebel copyright information.
About SRF	N/A	Opens a dialog box that displays information about the most recent full incremental compile.
About Visible Views	N/A	Displays the list of views in the repository and if each view is visible. Visibility for each view depends on the license key you use when you install Siebel Tools. For more information, see *Description of the About Visible Views Dialog Box*.

Menu Item (Siebel Tools)	Menu Item (Web Tools)	Description
Using Help	N/A	Opens *Siebel Tools Online Help*.

Description of the About Visible Views Dialog Box

The following table describes the information that the About Visible Views dialog box shows. To view this dialog box, click Help and then click About Visible Views.

Column	Description
View Name	Displays the view name. The views in this column are not the predefined views.
Visible	Indicates whether or not the view is visible in the application.
Application	Indicates whether or not the view is associated with Siebel Tools.
View	Indicates whether or not the view always appears as a view.
Responsibility	Indicates whether or not the view is associated with user responsibilities.
License	Indicates whether or not the view visibility depends on the license key that you use during installation.
Platform	Indicates whether or not the view visibility depends on the platform that is used.

Application Toolbars

This topic describes the application toolbars that are available in Siebel Tools.

- *Edit Toolbar*
- *List Toolbar*
- *History Toolbar*
- *Simulator Toolbar*
- *Format Toolbar*
- *WF/Task Editor Toolbar*
- *Configuration Context Toolbar*

Edit Toolbar

The following table describes the buttons that you can click on the Edit toolbar.

Button	Label	Description
📄	New	Starts the New Object Wizard that allows you to create applets, views, charts, and other objects.
💾	Save	Saves modifications in the current editing window if you use Layout, Menu, or Basic Scripts.
💾	Save All	Saves modifications in all open editing windows.
✂	Cut	If you use this button while the cursor is in a property that contains text, then the text you choose is copied to the clipboard and the existing text is selected. In the Applet Designer, it copies the control you choose to the clipboard and deletes the existing control.
⎘	Copy	If you use this button while the cursor is in a property that contains text, then the text you choose is copied to the clipboard. In the Applet Designer, it copies the control you choose to the clipboard.
📋	Paste	Inserts text from the clipboard to a text property at the insertion point. In the Applet Designer, it inserts a control from the clipboard.
↶	Undo	If you have not saved the object, then this button reverses the last modification you made to a property value in the Object List Editor or Property window.
↷	Redo	Reapplies modifications after the Undo command runs.

List Toolbar

The following table describes the buttons that you can click on the List toolbar.

Button	Label	Description
▤	Add New Record	Creates a new object in the Object List Editor and positions the cursor in the first required property.
⏮	First Record	Goes to the first object in the list.
◀	Previous Record	Goes to the object preceding the current selection.
▶	Next Record	Goes to the object succeeding the current selection.
⏭	Last Record	Goes to the last object in the list.
🔍	New Query	Allows you to specify one or more filters on the set of objects that the Object List Editor shows.
🔍	Execute Query	Runs the query you specify. This button does the same as pressing ENTER.
A Z ↓	Sort Ascending	Modifies the order that is used to display objects. It sorts them in ascending order according to the currently chosen property column.

Button	Label	Description
Z A ↓	Sort Descending	Modifies the order that is used to display objects. It sorts them in descending order according to the currently chosen property column.
▽	Filter Version	Displays only the most recent version of each Workflow Process or task UI that the Object List Editor shows.

History Toolbar

The following table describes the buttons that you can click on the History toolbar.

Button	Label	Description
⬅	Go Back	Returns to the previously displayed screen.
➡	Go Forward	Returns to the subsequent displayed screen.
+	Add Bookmark	Displays the Add Bookmark dialog box that allows you to create a bookmark to the object you choose. You can use this menu item to create a *bookmark*, which is a feature that allows you to return directly to an item.
≡	Bookmark List	Displays the Bookmarks dialog box that allows you to choose an existing bookmark. You can also use this dialog box to rename or delete an existing bookmark.

Simulator Toolbar

Note: This information applies to Siebel Tools and Web Tools.

The following table describes the buttons that you can click on the Simulator toolbar.

Button	Label	Description
▶	Start Simulation	Starts the simulation of a Workflow Process.
❙▶	Simulate Next	Simulates the next Workflow Process step.
⏭	Complete Simulation	Completes the simulation of a Workflow Process.
🟥	Stop Simulation	Stops the simulation of the Workflow Process.

Format Toolbar

The following table describes the buttons that you can click on the Format toolbar.

Button	Description
	Aligns the near edges of controls.
	Aligns the centers of controls along a vertical axis.
	Aligns the far edges of controls.
	Aligns the highest points of controls.
	Aligns the middles of controls along a horizontal axis.
	Aligns the lowest points of controls.
	Makes the controls the same width.
	Makes the controls the same height.

Button	Description
	Makes the controls the same size.
	Makes the horizontal spacing between controls equal.
	Increases the horizontal spacing between controls.
	Decreases the horizontal spacing between controls.
	Removes the horizontal spacing between controls.
	Makes the vertical spacing between controls equal.
	Increases the vertical spacing between controls.
	Decreases the vertical spacing between controls.
	Removes the vertical spacing between controls.
	Centers the controls vertically.
	Centers the controls horizontally.
	Aligns the labels to the near margin.
	Centers the labels.
	Aligns the labels to the far margin.

Configuration Context Toolbar

The following table describes the items that are available on the Configuration Context toolbar.

Drop-Down List	Description
Target Browser	Allows you to choose a target browser for layout editing and for scripting.
Application	Allows you to configure objects for a specific Siebel application. Typically, you use All Applications. If you choose a single application from the list, then you can configure objects, such as applets or views, to show or behave differently for only the application you choose.
Variable	Allows you to specify a display style for an applet for previewing, such as parent, child, or grandchild. An applet might be displayed differently depending on the underlying Web template. For example, an applet header might not appear if it is displayed as a grandchild.

Dialog Boxes That Compare Objects

This topic describes the dialog boxes that you use to compare objects.

Elements of the Compare Objects Dialog Box

The following table describes the elements of the Compare Objects dialog box.

Element	Description
First Selection Section	Displays the object hierarchy as a tree. To expand a tree, you can use the controls in the First Selection window or the Second Selection window. For example, if you expand the tree in the First Selection window, then the following also happens: • Expands the tree in the First Selection window. • Expands the tree in the Second Selection window. • Displays the child object types that each parent object contains. • Displays a dashed line to represent a child object that does not exist in an object.
Second Selection Section	See description for First Selection Section.
Properties Section	Displays the properties that are different.
Display Section	Determines how items are displayed in the Compare Objects dialog box. The following options are available: • **Show All Objects.** Displays all child objects in the First Selection section and the Second Selection section. • **Show All User Properties.** Displays all user properties in the Properties section. • **Show System Properties.** Displays system properties in the Properties section (for example Created, Created By, Updated, and Updated By).

Element	Description
➡	Synchronizes objects from the repository that the First Selection section represents with the repository that the Second Selection section represents. For more information, see *Comparing and Synchronizing Objects Between Repositories and Archives*.
⬅	Synchronizes objects from the repository that the Second Selection section represents with the repository that the First Selection section represents. For more information, see *Comparing and Synchronizing Objects Between Repositories and Archives*.
++	Expands the entire tree in the First Selection section and the Second Selection section.
--	Collapses the entire tree in the First Selection section and the Second Selection section.
Delete Button	Deletes objects after a comparison.

Dialog Boxes That Set Development Options

This topic describes the dialog boxes that you use to set development options. It includes the following information:

- *Development Options for Visualization Views*
- *Development Options for Scripting*
- *Development Options for Debugging*

Development Options for Visualization Views

The following table describes the development options that you can set for visualization views. For information about using this dialog box, see *Viewing Object Relationships*.

Option	Description
Use System Font	Uses a system font for the visualization views.
Use a Custom Font	You can use the Font, Size, and Zoom drop-down lists to choose your preferred font.
Boxes with 3D borders	Displays boxes with a three dimensional border.

Option	Description
Icon and name only	Displays the object name and the same object icon that the Object Explorer shows.
Simple outline boxes	Displays each object name in a simple box.
Always print outline style	Displays visualization details in outline style.

Development Options for Scripting

The following table describes the development options that you can set for scripting. For more information, see *Setting Options for Siebel Script Editor*.

Section	Option	Description
Font	Name	Sets the font name that appears for a script.
Font	Size	Sets the font size that appears for a script.
Script Assist	Enable Method Listing	Drop-down list that includes the methods and properties that are available for a declared object.
Script Assist	Tab Width	Specifies the number of spaces that will be used for a tab character in the script. The default value is four spaces.
Script Assist	Enable Auto Complete	If this check box is selected, then a method name or property name will be auto completed. It does this if you enter the minimal number of unique characters that are required to complete the name and press **`Ctrl+Space`**. If strings are found that are not unique, then it displays a drop-down list.
Script Assist	Auto Indent	If this check box is selected, then each succeeding line of script will be indented to the position that the current line sets.
Script Assist	Enable Favorites	If this check box is selected, then the object, method, or property name that you use most frequently will appear in italics at the start of the Script Assist window.
Script Assist	Engine Settings	For more information, see *Setting Options for eScript* .
Languages	Default Language for New Scripts	You can choose eScript or Visual Basic from the drop-down list.
Debugging	Debugging Settings	Allows you to set options for Siebel Server Script Debugger. For more information, see *Setting Debug Options for Siebel Server Script Debugger* and *Setting Debug Options to Open the Siebel Client*.

Development Options for Debugging

The following table describes the development options that you can set for debugging. To access this dialog box, click the View menu, click Options, and then click the Debug tab. The settings you make on the Debug tab are stored in the following user preference file:

```
loginId&SiebelTools.spf
```

It stores this file in the `$SIEBEL_HOME\BIN` folder.

Option	Description
Executable	Enter the name of the Siebel Web Client executable. For example: `siebel.exe` The default value is siebel.exe. This executable is run in debug mode or automatically after compile finishes.
CFG File	Enter the name of the configuration file that the Siebel client uses. For example: `C:\Program Files\Siebel\8.0\web client\BIN\ENU\uagent.cfg`
Browser	Enter the path to the browser executable. For example: `C:\Program Files\Internet Explorer\iexplore.exe`
Working Directory	Enter the Siebel root folder. This folder includes the Siebel executable and the DLLs (dynamic link libraries). For example: `C:\Program Files\Siebel\8.0\web client\BIN`
Arguments	Enter the options that will be used when the watch window opens: • `/h`. Enables local debugging of server scripts. • `/s file name`. Enable SQL spooling.
Prompt for This Information Each Time	If this check box is selected (contains a check mark), then information will be displayed each time a debug operation is run. For example, the name of the executable, the name of the CFG file, the browser configuration, and so on will be displayed.
Show Workflow Primary Business Component Data	If this check box is selected (contains a check mark), then the Watch window in the Workflow Simulator displays information about the Workflow Process. It displays the name of each business component field and the value for each of these fields. These fields include fields from the primary business component of the business object that the Workflow Process references.
User Name	Enter the user name that Siebel CRM requires to log in to the Siebel application you are debugging.
Password	Enter the password that Siebel CRM requires to log in to the Siebel application you are debugging.

Option	Description
Data Source	Choose a default data source. The values you can choose depend on the configuration file that you specify in the CFG File option. The Siebel Web Client connects to this local database.

Parameters That Convert Symbolic Strings

This topic describes the parameters that you can use to convert a symbolic string. You use these parameters with a conversion utility. For more information, see *Converting Symbolic Strings*.

This topic includes the following information:

- *Conversion Export Utility Parameters*
- *Conversion Import Utility Parameters*

Conversion Export Utility Parameters

The following table describes the parameters that you can use with the conversion export utility.

Parameter	Description
Filename	Required. Specifies the name of the export file.
Repository	Required. Specifies the name of the repository. The repository name is case sensitive.
Object	Required. Specifies the object type that contains the strings that this utility exports. For example: `Control` The object name is case sensitive.
LogFile	Specifies the name of the log file.
Language	Specifies the language that this utility uses as the primary language to match when it searches for duplicate symbolic strings. For example, each symbolic string includes the following child records: • English (ENU) • French (FRA) • German (DEU) If you set the Language parameter to ENU, then the conversion export searches for matches between the ENU records. If it finds matches, then it examines the other child records of the other languages. If all child records match, or if one language includes a superset of one of the other languages, then this utility considers them as matching symbolic strings.
MatchMin	Specifies the minimum number of matches in a set of matching symbolic strings before this utility writes them to the file. The default value is 2.

Parameter	Description
SQLLog	Specifies the SQL log file name. If you set this parameter, then this utility logs all SQL that it runs to this file.
ExcludeNull	Specifies a TRUE or FALSE value. If TRUE, then this utility excludes null values for conversion consideration. The default value is TRUE.
UseFullMatch	Specifies a TRUE or FALSE value. If TRUE, then this utility matches records against all the other possible match candidates before it discards them. The default value is TRUE.
UseExactMatch	Specifies a TRUE or FALSE value. If TRUE, then this utility considers records as a match only if all of the records include the same number of language records, and if the same values exist for each language. It does not consider partial matches. The default value is FALSE.
SkipInactive	Specifies a TRUE or FALSE value. If TRUE, and if the Inactive property of the record is set to Y, then this utility skips this record. The default value is TRUE.

Conversion Import Utility Parameters

The following table describes the parameters that you can use with the conversion import utility.

Parameter	Description
Filename	Required. Specifies the name of the import file. You must use the same name that you use when you export symbolic strings.
Repository	Required. Specifies the name of the repository.
LogFile	Specifies the log file.
UnlockProjects	Specifies a TRUE or FALSE value. If TRUE, then this utility unlocks all projects when the conversion finishes. This configuration is useful if multiple instances of the conversion service run against the same database. The default value is TRUE.
SkipParentUpdates	Specifies a TRUE or FALSE value. If TRUE, then this utility does not update parent objects to use the symbolic string. The default value is FALSE. Set SkipParentUpdates to TRUE only if you do not simultaneously run multiple instances of the import. If you set SkipParentUpdates to TRUE, and if you run multiple instances, then errors might occur. The utility might abort an update or delete records because another instance updates the project at the same time.
SQLLog	Specifies the log file name. If you use this parameter, then this utility logs all SQL that it runs to the file you specify.
Project	Required. Specifies the name of the project in the repository that includes the new strings. Predefined symbolic strings reside in the Symbolic Strings project. You can configure this utility to import custom strings. For more information, see *About Predefined Objects*.

Parameter	Description
DeleteLocales	Specifies a TRUE or FALSE value. If TRUE, and if all translatable strings are NULL, and if language override is not enabled, then this utility deletes locale records. If FALSE, then this utility sets the locale record to Inactive. The default value is TRUE. For more information, see *Setting the Language Mode*.
CheckTranslateFlag	Specifies a TRUE or FALSE value. If TRUE, and if the Translate property for the object is N, then this utility does not convert this object. The default value is TRUE.
LogErrorRecords	Specifies a TRUE or FALSE value. If TRUE, then this utility exports all error records to a separate log file. The default value is FALSE.

14 Glossary

Advanced Compile

A feature that you can use to assist with localization.

Applets window

A window that displays information about a view and allows you to add applets to that view.

Application Deployment Manager (ADM)

A feature that you can use to administer a Siebel CRM deployment.

bookmark

A feature that allows you to return directly to an item.

Bookmarks window

A window that allows you to navigate directly to an object that you use frequently.

breakpoint

A marker in a line of code that stops code from running at that line.

canvas

A background that appears in different designers.

collection function

A type of function that includes a finite set of values.

compound query

A type of query that locates records according to more than one condition.

conflict

A situation that occurs if an object that you are attempting to import from an archive file does not match the corresponding object in the repository.

Controls/Columns window

A window that displays the controls and list columns that you can add to an applet layout if you use the Applet Web Template Editor.

custom object

A new object that you create.

declarative configuration

A type of programming technique that uses objects and object properties in the Siebel Repository to implement the logic that your business requires.

developer conflict

An error that occurs if two separate groups or developers update the same object.

full get

A type of Get that copies all projects from the server repository to your local repository.

get

The act of copying a project from the server repository to your local repository.

hidden object type

An object type that Siebel CRM does not show in the Siebel Web Client.

hotfix

A feature you can use to quickly update the production environment.

Incremental Repository Merge

A feature that allows you to merge multiple repositories and apply patches.

language mode

A mode that allows you to configure Siebel CRM to display text in a language other than English.

language override

A nontranslatable locale property that you can configure differently for different locales.

Locale Management Utility (LMU)

A utility that you can use to manage how you configure Siebel CRM to localize text strings, such as field labels, and other locale properties, such as the height and width of controls.

mid-level release

A type of release that includes objects you modify from a time frame that you specify.

modified object

A predefined or custom object that you modify.

new object wizard

A feature you can use that guides you through the steps of creating a new object.

object definition

Implements one piece of the software. It consists of object properties, which are characteristics of this piece of software.

Object Explorer

A window that displays the Siebel object hierarchy.

Object List Editor

A window in that displays the object definitions of the object type that you choose in the Object Explorer.

object property

A characteristic of a piece of software.

object tagging

A version control feature that Siebel uses to associate a repository modification with a tag or group. You can use it to export all the work that a group of developers performs.

object type

An entity that includes a predefined set of properties.

Palettes pane

A pane containing items, which you can add to an object.

parent and child relationship

A type of hierarchical relationship between one object type and another object type.

predefined object

An object that comes already defined when you first install Siebel CRM.

predefined symbolic string

A string that comes predefined with Siebel CRM.

prior standard repository

The repository that comes predefined with Siebel CRM. Siebel uses it to determine if you modified any object in the repository since the prior release.

project

An object type that your development team can use to help make sure only one developer works on an object at one time.

property value

Information that you enter into the column of an object definition.

Properties pane

A pane that displays the properties and the property value for the object that you choose in the Object List Editor.

pseudolocalization prefix

A prefix that Siebel uses to test the appearance of string in a language.

reference repository

A prior version of the repository.

right-click menu

A type of context-sensitive menu that appears if you right-click an object or an element.

Siebel Application Upgrader

A utility you can use to get new features from the latest software release while preserving the custom configuration you created in the current repository.

Siebel Delta File (SDF)

A type of file that is similar to an SIF file except that an SDF file contains only information about modifications to an object.

Siebel Repository

A set of tables that includes Siebel objects and server scripts.

Siebel Repository patch file (SPF)

A type of file that includes exported objects.

Siebel Script Editor

An editor that allows you to create and maintain scripts that use Siebel VB, Siebel eScript, or Browser Script.

Siebel Web Tools

An integrated development environment that you can use to configure Siebel CRM.

simple query

A type of query that locates records according to one condition.

symbolic string

An object that you can use to store the value of a string.

symbolic string reference

A reference to a symbolic string.

top-level object type

An object type at the start (or highest level) of the object hierarchy.

Touch

A version control feature in Siebel that allows you to tag an object even if you do not modify this object.

translatable string

A type of string that Siebel CRM can translate to another language.

Type deduction

A feature of the Siebel eScript that determines the type of local variables that a script uses.

Validate Tool

An error correction tool you can use to validate the semantic consistency of an object.

Web template editor

An application external to Siebel that allows you to edit a Web template.

Web Template Explorer window

A window that displays the HTML code of a Siebel Web Template.