Siebel Reports Administration Guide

Version 7.7, Rev. A

June 2004

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

Printed in the United States of America

PRODUCT MODULES AND OPTIONS. This guide contains descriptions of modules that are optional and for which you may not have purchased a license. Siebel's Sample Database also includes data related to these optional modules. As a result, your software implementation may differ from descriptions in this guide. To find out more about the modules your organization has purchased, see your corporate purchasing agent or your Siebel sales representative.

Proprietary Information

Siebel Systems, Inc. considers information included in this documentation and in Siebel eBusiness Applications Online Help to be Confidential Information. Your access to and use of this Confidential Information are subject to the terms and conditions of: (1) the applicable Siebel Systems software license agreement, which has been executed and with which you agree to comply; and (2) the proprietary and restricted rights notices included in this documentation.

Contents

Chapter 1: What’s New in This Release

Chapter 2: Understanding Your Reports Development Environment

Chapter 3: Reporting in the Siebel Web Clients

Chapter 4: Managing the Siebel Reports Server

Chapter 5: Global Report Modifications

Chapter 6: Creating a Simple List Report

Chapter 7: Reports with Group Sections

Chapter 8: Master-Detail Reports

Chapter 9: Composite Datastreams

Chapter 10: Sorting Records in Memory

Chapter 11: Using Graphics in Reports

Chapter 12: Smart Reports

Chapter 13: Parameterized Reports

Chapter 14: Developing Multilingual Reports

Chapter 15: Report Business Service

Appendix A: Library Reference

Appendix B: Method Reference

Appendix C: Smart Reports List of Values

1 What’s New in This Release

What’s New in Siebel Reports Administration Guide, Release 7.7, Rev. A

Table 1 lists changes in this version of the documentation to support Release 7.7 of the software.

Table 1. What’s New in Siebel Reports Administration Guide, Release 7.7, Rev. A

Topic	Description
Chapter 3, “Reporting in the Siebel Web Clients”	Graphics for reports generation in the Siebel Web Clients were added to the chapter.
“About Siebel Report Object Types” on page 24	Information about using the New Report Wizard is removed.
“Requesting Reports in the Siebel Web Client” on page 37	Progressive viewing allows the first page of a report to be displayed before the entire report is completed. Progressive viewing is also in the Siebel Dedicated Web Client.

What’s New in the Siebel Reports Administration Guide, Release 7.7

Table 2 lists changes described in this version of the documentation to support Release 7.7 of the software.

Table 2. New Features in the Siebel Reports Administration Guide, Release 7.7

Topic	Description
Installation instructions for the Reports Server are located in another guide.	Reports Server installation can be found in the *Siebel Installation Guide* for the operating system you are using.
Chapter 3, “Reporting in the Siebel Web Clients.”	Printing and scheduling of reports from the Siebel Web Clients are performed using Actuate Active Portal. RS APIs are no longer used.

21

2 Understanding Your Reports Development Environment

This chapter consists of the following topics:

About the Siebel Reports Server

Siebel applications ship with standard reports. To modify these reports or add new reports, you need to use Siebel Tools and Actuate e.Report Designer Professional as described in this topic.

You create and modify reports in two locations:

- In Siebel Tools, by creating and modifying Report or other object definitions and setting properties within them. These object definitions are executed at run time.
- In Actuate e.Report Designer Professional, by creating and modifying Report Object Design (ROD) files, which are then compiled and executed.

Although changes may be made only in Siebel Tools or in Actuate e.Report Designer Professional, frequently report redesign work requires making changes and additions in both places.

Siebel Tools modifications use the Report, Report Field, Report Locale, Sub Report, Sub Report Field, View Report, and View Report Locale object types. These modifications affect the following areas:

- Defining the structure of the data exported from the Siebel application to the Actuate report, which the Actuate report receives into its datastreams
- Attaching reports to the Reports menu for specific views

ROD file modifications in Actuate e.Report Designer Professional alter various classes and subclasses that define report behavior, appearance, data acquisition, and so on for one report.

Actuate e.Report Designer Professional is a visual design editor from which object-oriented Actuate BASIC code is generated and compiled from the report design (ROD file) and referenced library classes (ROL, or Report Object Library, files) into an executable report program. The resulting executable program is an ROX (Report Object Executable) file. When the executable program is run, the result is an instance file containing both report specifications and data. The instance file is in ROI (Report Object Instance) format, suitable for display in the Actuate Viewer on a Microsoft Windows client (Mobile Web Client). When the instance file is requested by a Web browser (directly from the Reports Server in Web client and dedicated Web client environments), it is converted to browser-specific DHTML format from the ROI. This is illustrated in Figure 1.

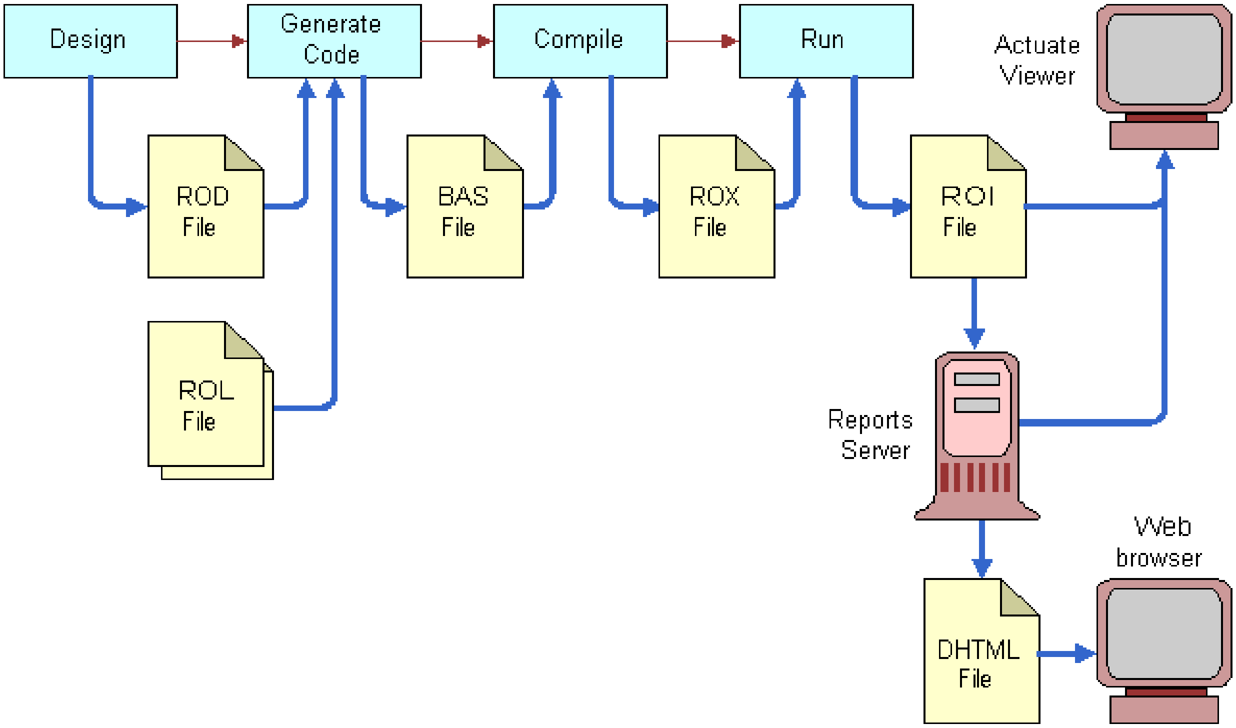


Figure 1. Actuate Report Generation and Display Steps

As shown in Figure 1, the ROI file generated by in the interactive mode can be sent directly to the Siebel Reports Server for long-term storage and availability. The ROI file is also accessed by Web browsers and thin clients. When in the Siebel Web Client, the user specifies whether immediate display or Reports Server processing is required when requesting the report.

About the Development Environment

This section summarizes certain features of the Actuate development environment as they apply to Siebel reports. The specifics of using Actuate e.Report Designer Professional are explained in greater detail in *Developing Advanced e.Reports,* in the Actuate documentation set.

About the Actuate File Types

Actuate uses or generates files of the following nine types:

- **ROD (Report Object Design).** An ROD file is a report layout file. An ROD file exists for each standard report, and you create a new ROD file for each new report you create. The ROD files for the standard reports are provided with Siebel Tools.
- **ROL (Report Object Library).** An ROL file is a library file. A library file contains reusable components you can add to design files. Since you can subclass, copy, or reference objects from libraries, you usually can reuse existing ones, and you do not have to understand how to construct them.
- **BAS (BASIC source code).** A BAS file is generated during the build and compilation processes. It is generated from the ROD file being compiled and from all included library modules from ROL files. It is an intermediate file format used in the subsequent compilation step and is not directly modified by the developer. A BAS file can also be used to implement reusable Actuate BASIC routines for inclusion in report designs.
- **ROX (Report Object Executable).** An ROX file is an executable report, that is, a compiled ROD file. When you run a report, the Siebel application executes the ROX file. Note that Siebel applications include ROX files for the standard reports. When you customize standard reports or create new reports, you need to replace the corresponding ROX files or add them to the appropriate directory to make them available to the Siebel application.
- **ROI (Report Object Instance).** An instance file is what the user sees when the report is running in the Actuate window in the Siebel application. You interact with an instance file only when saving a report or when viewing it using an external viewer (Siebel Mobile Web Client mode).
- **ROV (Report Parameter Values).** The parameter values file contains a report request's parameter definitions and values. Actuate creates the parameter values file automatically when users issue a request. The user enters parameter values in the Application screen (see parameterized reports documentation) then the information is written to a parameter values file, which can then be used to generate a report based on the specified parameter values.

  NOTE: If a control is not seen properly, or seems to have disappeared in the ROI file, review how the control is placed in the ROD file. Check the positioning, the size, and whether the control is overlapping another control. However, even if the control has the correct position or size, it still may not be seen properly in the ROI file.

- **DHTML (Web page).** A DHTML file is what the user sees when the report is obtained from the Reports Server using a Web browser (Siebel Web Client mode).

About the Directory Structure for Siebel Reports

Actuate files in the Siebel environment reside in the following subdirectories of the Siebel development directories:

- **Executables Directory.** \Siebdev\REPORTS. Report executables (ROX files) must appear in this directory to run. In the typical installation, this is `C:\Program Files\siebel\7.7\Tools\REPORTS`. If multiple languages are supported, a separate subdirectory of \REPORTS is provided for each and is identified by its language code, such as \Siebdev\REPORTS\DEU or \Siebdev\REPORTS\FRA.
- **Development Directory.** \Siebdev\RPTSRC. \RPTSRC is the development directory; it holds ROD and ROL files. In the typical installation, this is `C:\Program Files\siebel\7.7\Tools\RPTSRC`. It is divided into subdirectories by language. New reports that you create should be placed here, in the \language_code subdirectory. Each language subdirectory is further divided into \STANDARD and \LIB, such as \Siebdev\RPTSRC\ENU\STANDARD and \Siebdev\RPTSRC\ENU\LIB.
- **Standard Reports Directory.** \Siebdev\RPTSRC\STANDARD. An example is `C:\Program Files\siebel\7.7\Tools\RPTSRC\STANDARD`. This subdirectory is where the design files for the standard reports are located. Do not place customer reports in this directory, create a subdirectory called `\custom` or something similar.
- **Libraries Directory.** \Siebdev\RPTSRC\LIB. An example is `C:\Program Files\siebel\7.7\Tools\RPTSRC\LIB`. The datastream (source data definition) files are located here. These are generated from within Siebel Tools. This folder also contains the library files that are used by all Siebel reports, such as sssiebel.rol, sscustom.rol, and so on. As with Standard reports, do not place custom reports in this directory, create a subdirectory called `\custom` or something similar

You develop the ROD file in the standard reports directory or the development directory, depending on whether you are modifying a standard report or creating a new report. You compile the finished design in Actuate e.Report Designer Professional and move the resulting ROX file to the executables directory. Initially, you would deploy the ROX file on your own computer for testing. When it is ready to be deployed, the ROX files goes to the appropriate folder on the iServer report encyclopedia volume.

About Actuate Libraries

Every Siebel report, whether standard or custom, includes the following libraries:

- **sssiebel.rol.** This is also known as the Siebel library. It is derived from afc.rol and contains all the base classes for the sscustom library. It is automatically included as a part of sscustom.rol. Do not modify the contents of sssiebel.rol.
- **sscustom.rol.** This is also known as the Custom library. It is derived from the sssiebel.rol library. You can make modifications in sscustom.rol to make global changes to fonts, headings, and so on that impact all reports. While making modifications is accepted, this file is not to be used as a scratch pad. Before making any changes to this library, make a backup copy of the out-of-the-box sscustom.rol file. Do not delete any files from this directory. You will frequently incorporate subclassed or copied objects from sscustom.rol into report design files you are creating or modifying.

- **<reportname>.rol.** This is your data supply library file; there is one for each Actuate report used by your Siebel application. For example, the ACLIST (Account List) report will have a datastream file called Aclist.rol. The data supply library file is automatically generated by Siebel Tools for the currently selected report object definition when you choose the Generate Actuate Report option from the Tools menu.

- **sssiebel.bas.** This is also known as the sssiebel BASIC file. It is a BASIC source code file containing methods used by all Siebel reports, especially for object and data interfaces between Siebel and Actuate. It is included in all standard reports by default and must be included in custom reports so that they work correctly for server reporting. Do not modify the contents of sssiebel.bas.

- **sscustom.bas**. This is also known as the Custom BASIC file. It is a BASIC source code file containing reusable Actuate BASIC routines for inclusion in report designs.

Figure 2 illustrates the inheritance structure of components in a report design file.

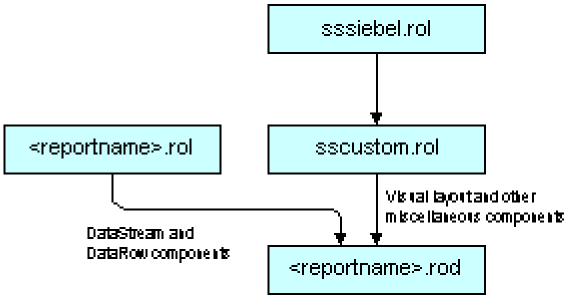

sssiebel.rol
<reportname>.rol
sscustom.rol
Visual layout and other miscellaneous components
DataStream and DataRow components
<reportname>.rod

Figure 2. Inheritance Structure of Components in a Siebel Report

The classes in the sssiebel.rol library are derived from the Actuate Foundation Class library. As shown in Figure 2, the classes in sscustom.rol are derived from sssiebel.rol classes. The sssiebel.rol library is a system layer providing a link between Actuate and Siebel applications. sssiebel.rol is reserved for Siebel product enhancements and should not be modified by a developer. The sscustom.rol library is provided for modifications by developers, although you will use most of its components unmodified in your report designs. You should never modify the sssiebel.rol library, only the sscustom.rol library.

NOTE: Components from afc.rol (and or components from the Actuate e.Report Designer Professional toolbar) and sssiebel.rol should not be used by report developers for designing Siebel-Actuate reports. When developing Siebel application reports, only use the sscustom.rol library.

About Actuate Design Files

An ROD file is a report design file. It defines the layout, structure, and behavior of a report. The ROD files for the standard Siebel reports are provided with Siebel Tools. A design file is modified in the main window of the Actuate e.Report Designer Professional software, called the Design Editor window. The design file for the Opportunity Summary report (opsum.rod) as it appears in the Design Editor is shown in Figure 3.

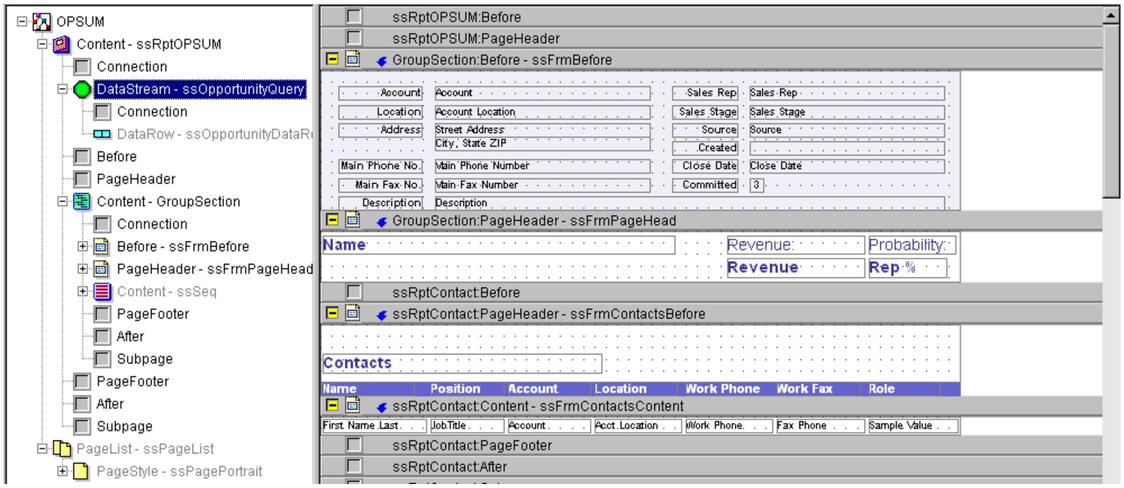


Figure 3. Report Design for a Siebel Report in the Design Editor

The Design Editor window consists of a structure tree on the left and a layout grid on the right. The structure tree is populated with components. The slots (also called nodes) that hold these components can be of various types according to their use and behavior in the report. Specialized icons identify the types of their corresponding components. Of particular interest are the following node types in the structure:

- **Report slot.** Identifies the current report. This is the top-level component (OPSUM in Figure 3).
- **Content slots.** Identifies where the content is coming from in some portion of the report and how it is laid out. A content node often contains page header, before, content, after, and page footer nodes, which correspond to frames (rectangular report areas) in which visual report elements can be laid out.
- **Datastream slots.** Identifies the source of data. In reports for Siebel applications, this is a set of rows corresponding to a current view; its columns are from one or more business components. The datastream node in the structure actually points to an external library file, which is generated from Siebel Tools based on the contents of report (and child) object definitions in Siebel Tools.
- **Pagelist slots.** Holds general page layout information for the report and generally is also obtained from an external library file that is common to all the reports.

Recall that Actuate is an object-oriented software product. Each of the icons in the structure tree represents an object, which can have child objects (as illustrated in the tree) and which has a properties list that you can edit. You may note the similarity to Siebel Object Explorer.

The class assigned to an object in Actuate determines its behavior. When new objects are created, or moved from one location to another, you need to be careful not to alter the relationship between an object and its current class, except when you are explicitly told to do so. The following three concepts are related to changing the relationship between an object and its class:

- **Subclassing.** Subclassing an object results in the creation of its own class. This class is based on and linked to the class of the original object. All updates to the original object are inherited by your new object, except for updates to the parts you have changed in the subclass. This technique is commonly used in Siebel reports so that your reports inherit new behavior as Siebel products are upgraded.
- **Referencing.** Referencing means referring to an existing component rather than creating a new component. You use a reference when an existing component meets your needs and you want to use it exactly as it is, no matter how it might change in the future.
- **Copying.** You can copy a component from one part of a report design to another. After you copy a component, it has no relationship to the component from which it was copied.

See the information on concepts in *Developing Advanced e.Reports* in the Actuate e.Report Designer Professional documentation for a review of Actuate concepts.

How the Siebel Application and Actuate Interact

In the Siebel Web Client, the report instance is obtained by the client in DHTML rather than ROI format, and the report is viewed using a Web browser window. In the Siebel Mobile Web Client, the Actuate Viewer is called the Siebel Report Viewer.

NOTE: The Siebel Report Viewer does not allow configuring of the menu items on the actual viewer.

The current business object (obtained from the view) and the current query create a context that determines which data is sent to the report. In Mobile Web Client reporting mode in Windows, the data is passed through variables across the interface between the Siebel application and the embedded Actuate viewer, and it is accessed using methods in the report design. In Web Client reporting mode, the object interface is between the Object Manager and the Reports Server.

When the user is running a report from the Siebel Mobile Web Client, a corresponding report executable (ROX file) is invoked locally and the resulting instance (ROI file) is displayed. When a user is running a report from the Siebel Web Client or scheduling a report from the Siebel Mobile Web Client, the report executable is invoked on the Reports Server at the appropriate time, and the instance is stored on the Reports Server. It may also be obtained automatically from the Reports Server for local display, depending on the environment and the user’s request.

Running a Siebel Report from Actuate e.Report Designer Professional

You can run a Siebel report from Actuate e.Report Designer Professional connected to a Siebel Dedicated Web Client. You can see the run-time behavior of the Siebel report, set break points, and so on.

To run a Siebel report from Actuate e.Report Designer Professional

1. If a Siebel Dedicated Web Client is running on your desktop, close it. Open the Actuate e.Report Designer Professional software.
2. Choose File > Open and navigate to Aclist.rod in \Siebdev\RPTSRC\ENU\STANDARD.

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

   See the *Global Deployment Guide* on the *Siebel Bookshelf* for a list of three-letter International Standards Organization (ISO) language extensions.
3. Take the following action:
   a. Click the Run button to compile and execute the report.

      You might see an error message, such as:

      ```
      OLE Server couldn't be started.
      ```
   b. Click OK.
4. Close this error message box and halt the report execution.
5. Open Siebel Sales Dedicated Web Client from the Start menu.
6. From the application-level menu, choose Navigate > Site Map > Opportunities > My Opportunities.

   The wrong view is intentionally being opened to demonstrate the result.
7. Return to Actuate e.Report Designer Professional with the Siebel application still open.
8. Run the report again by clicking the Run button.

   This time you see a different message, such as:

   ```
   There is no data to display.
   ```
9. Close the Actuate Output message box and return to Siebel Sales.
10. From the application-level menu, choose Navigate > Site Map > Accounts > My Accounts.
11. From the Show drop-down list, choose My Accounts.
12. Return to Actuate e.Report Designer Professional and click the Run button.

    This time, the Account List report should run successfully.

This exercise demonstrates that a Siebel report does not run locally in Actuate unless two conditions are met:

- A Siebel Dedicated Web Client application is running, making the Siebel software available as an OLE container.
- The current view in the application is based on the correct business object for supplying data to the report being run. In the case of the example, a report listing account data requires that a view based on the Accounts business object be active.

Since a user executes Siebel reports only from the views in which they are designed to be run, these kinds of errors do not occur in a properly configured Siebel application.

Note that when the user restricts the data in the view with a query, only the data meeting the query constraints appears in the report. The Siebel application passes only the data from the current query to the Actuate viewer.

About Run-Time Report Parameters

Table 3 describes the parameters passed through the report request to the Siebel Object Manager when the report is initiated. The report will not run if parameters are not defined correctly. For more information, see "baseReport" on page 236.

Table 3. Report Parameters

Report Parameter	Description
ParamLocale	Language Code. The report language code chosen from the User Preference view.
ssActiveRowId	Active row-id of primary business component for the report. Used for current-record-only reports.
ssBookmark	Contains information about business component state, including queries, when the report request is submitted. Used only for Web Client reports.
ssBusObjectName	Name of the business object corresponding to the active view.
ssDataLanguage	Language Code. The report language chosen from the User Preference view.
ssLanguage	Language Code. The language code of the application.
ssLocale	Locale Code. The report locale code chosen from the User Preference view.
ssOLEServer	Specifies the OLE server to connect to in order to obtain data displayed in the report. In version 7.0, OLE server is TWSiebel.SiebelWEBApplication.1 for interactively generated reports in Mobile Web Client and Siebel Report Server Access for server-based reports.
ssPassword	Siebel password.
ssPositionId	The current position Id of the user.
ssSearchSpec	Search specification. Set in Tools or in the report design.

Table 3. Report Parameters

Report Parameter	Description
ssSiebelSever	Siebel Server connect string. Used for the client to establish connection with the Siebel Server.
ssSortSpec	Sort specification. Set in the report design.
ssUserName	Siebel user name.
ssViewMode	Used for setting view mode on the Siebel Server side during report generation. Based on current view from which report is submitted or value set using Tools.

About Data Definition for Reports

The ROD file for a Siebel report must reference its datastream components from an ROL file, rather than through the data source and query definition processes used for non-Siebel reports in Actuate. This is because the Actuate viewer is obtaining data through the Siebel object interface rather than by directly accessing the database. This is consistent with Siebel standards for making sure that data access is always at the business object level, rather than data object level.

The structure of the exported data must be consistent between the Siebel application and the Actuate executable so that the data will be usable by the report. To accomplish this, the structure of the data for each report is defined in Siebel Tools using a report object definition and its children. It is then exported to a datastream file in ROL format using the Generate Actuate Report option in the Tools menu in Siebel Tools. The name of the ROL file to be generated is specified in the Template Name property in the report object definition in Siebel Tools.

The relationships between the names of the data supply ROL, report design, and executable files are explained in Table 4.

Table 4. Relationships Between ROL, ROD, and ROX Filenames

File	How Used	Where Name Is Specified
Data supply (ROL) file	Generated from Siebel Tools and loaded into ROD file as an included module.	Template Name property of the report object definition.
Report design (ROD) file	Specifies the layout and behavior of the report; subsequently compiled into an ROX file.	Identified to the Siebel application using the Access Base DB Name property of the report object definition. Originally created in Actuate e.Report Designer Professional.
Executable (ROX) file	Runs the report when executed.	Automatically receives the same name as the ROD, except for the filename extension.

The generated ROL file, by convention, has the same name as the ROD file into which it is intended to be incorporated. For example, the Opportunities - Summary report object definition specifies the name OPSUM in both the Template Name and Access Base DB Name properties, and the corresponding report design file is Opsum.rod. When a data supply library file is exported from Siebel Tools for this report, it is given the name Opsum.rol because of the Template Name setting. When the ROD file is compiled, the resulting executable is given the name Opsum.rox by Actuate e.Report Designer Professional. When the Siebel application invokes the executable from the view, the file it invokes is Opsum.rox because of the Access Base DB Name setting.

NOTE: It is not required that the data supply ROL filename match the filename of the ROD file into which it is incorporated. However, it is good design practice to have a separate data supply ROL file for each ROD file and to match the names where possible.

About Data Supply ROL Files

All ROL files contain reusable components that can be subclassed into design files in Actuate e.Report Designer Professional. For example, a report design subclasses design elements such as label, text, and frame controls from sscustom.rol, as described in Chapter 5, "Global Report Modifications." The incorporation of a data supply library file into the corresponding design file is a special application of this subclassing methodology. A report design subclasses the datastream component from the corresponding data supply ROL file.

The datastream component contains methods for accessing the necessary report data from the correct business object through the Siebel object interface. One or more data row components are defined for use by the datastream, each specifying the list of fields for a business component whose records are to be retrieved. The logic for the datastream component fetches and deletes instances of the data row until all records in the current query (and subqueries, if applicable) have been obtained and processed into the report.

Viewing the Contents of a Datastream Component

You can view the contents of the datastream and data row components in a report design using the Method Editor and the Component Editor.

To view the contents of a datastream component

1. Open a standard Siebel report in Actuate e.Report Designer Professional, for example Actlist.rod in C:\Siebdev\RPTSRC\STANDARD.
2. Expand the main report section component and double-click the datastream component.

   The Component Editor window appears.
3. Click the Methods tab.
4. Choose Start in the method picklist at the upper left in the window.

   The text for the datastream's Start method appears in the Method Editor window.
5. Examine the code for this method. Notice that this code activates each exported field in the business component.

6. Select Fetch in the Method picklist and examine the code for the Fetch method. Notice that the code obtains the value for each exported field.

Viewing the Contents of a Data Row Component

The data supply library file for a report is loaded into the report design file by invoking the Tools > Library Organizer menu option in Actuate e.Report Designer Professional. Its datastream and data row components subclass the AcDataSource and AcDataRow components in the Actuate Foundation Class library, respectively. The methods in the data supply library file override corresponding methods in the foundation class library. The contents of these methods are generated code, produced from the list of report fields and other report object definition properties and children in Siebel Tools.

To view the contents of a data row component

1. If you have not already opened a Siebel report, follow Step 1 on page 23.
2. Expand the datastream component in the component tree and select the data row component.
3. Right-click the data row component and select Properties in the pop-up window.

   The Component Editor window opens.

4. Click the Variables tab.

   The list of variables defined in the data row appears.

   Notice that all the variable names are in black, rather than gray, except for RowNumber. This indicates that these variables are defined locally in the class code, rather than inherited from the parent AcDataRow class.

   Notice too that the names of the variables are modified field names from the business component. The prefix ss has been added to each, and spaces and special characters have been replaced with underscore (_) characters.

The automatic definition of a set of variables, corresponding to the list of exported report fields, is the role of part of the generated code in the data supply library.

About Siebel Report Object Types

The following object types are used in Siebel Tools to define the structure of the data for each report and are used to generate the data supply ROL file for that report:

- **Report object type.** A report object definition provides the high-level properties for one report. Report properties identify the filenames of the generated data supply library and the report executable, the business component name, the report type (Actuate or Access), and so on. The property types:
  - **Business Component property.** This specifies the business component whose data is used in the main report. The business component of the subreport is specified in the subreport object definition.

  - **Template Name property.** This is the name of the data supply library generated in Siebel Tools (without the ROL extension) that supplies information to Actuate about the report. This property is left blank for Access reports. It is also left blank for custom Actuate reports in which the data supply library file is too complex to generate from Siebel Tools, and must be created as program code.
  - **Access Base DB Name property.** For Actuate reports, this is the name of the executable file (without the ROX extension) that the Siebel application will run when the report is selected. For Access reports, this is the MDB (database) file that contains the report specification.
  - **Class property.** The CSSActuateReportViewer class identifies an Actuate report.
  - **Search Specification property.** The conditional expression used to retrieve a subset of records. If a search specification is defined in the report object definition and you want to run the report from Actuate e.Report Designer Professional, the search specification does not pass through from Siebel Tools. To check if the search specification is working, you will have to run it from the Siebel application.

    NOTE: The same result occurs if the Current Record Only property (a type of search specification) is defined. Setting this property to TRUE does not pass through to Actuate e.Report Designer Professional from Siebel Tools.

    Also, if you set a search specification on your parent datastream in Actuate Start(), this search spec will override the dynamic view query results in the report and look through all the records in the business component.
  - **Sort Specification property.** You can set a sort specification in a report object definition to send the rows to Actuate through the datastream in sorted order. Otherwise, the rows are sent in the sort order on the current view. Ordering the data coming from the Siebel application will, at a minimum, improve performance when the report runs, and may be required to make the report work.

    NOTE: Setting a search or sort specification property is also possible for subreports.
  - **Menu Text property.** The menu text to appear in the Reports menu when this report is included in the active view by means of a view report object definition.
  - **Parameter Applet property.** The name of a parameter entry pop-up applet created in Siebel Tools.

- **Report Field object type.** A report field object definition identifies one field to be included in the report from the report's business component. In Actuate reports, the list of report field object definitions is incorporated into the data supply library file when it is generated. For a field to appear in the datastream, and hence in the report when it is run, it must be included as a report field object definition for the report in Siebel Tools. You can list more report fields than are actually used in the report. Note, however, that the system retrieves all listed fields, so listing unused fields needlessly degrades report performance. Each field in the ROL file is a variable on the child data row component of a datastream.
- **Report Locale object type.** A report field object definition that identifies the language-specific overrides associated with the Report object type.

- **Sub Report object type.** A subreport object definition that uses data from a detail business component and contains information to manage a detail portion of a master-detail report. One datastream component is included in the data supply ROL file for the main report and for each subreport child of the report. Each detail datastream component is generated from a subreport object definition and its subreport field children.
- **Sub Report Field object type.** A subreport field object definition identifies one field to be included in the subreport from the subreport's business component. The list of subreport field object definitions goes into the subreport datastream component as variables in the data row.

The following child object types of the View object type are used to attach a specific report, as defined in its report object definition, to a specific view:

- **View Report object type.** A view report object definition creates an association between a report object definition and a view, causing that report to be available in the Reports menu when the view is active.
- **View Report Locale object type.** A view report object definition that identifies the language-specific overrides associated with the View Report object type.

  NOTE: In Actuate, an apostrophe indicates that code after it will be read as comments. As a result, a field name in any of the report objects that contains an apostrophe will result in compilation errors.

In addition to report object types, virtual business components may be used to create reports. Virtual business components allow you to represent external data as a business component within a Siebel application. They also allow the use of business services to transfer data. For more information about virtual business components, see *Using Siebel Tools* and *Overview: Siebel eBusiness Application Integration Volume I*.

NOTE: For information on the set of properties for these object types, see object types on *Siebel Tools Online Help*.

Additional Siebel-Actuate Reporting Information

This section covers various additional information that pertains to reporting in Siebel applications.

About the Actuate e.Report Designer Professional and Siebel Tools Installation

Note the following points about software installation:

- Actuate e.Report Designer Professional is installed on a Windows client PC from the Siebel Windows client DVD-ROM. Follow the instructions in the *Siebel Installation Guide* for the operating system you are using.

  Installation of the Reports Server components is described in the *Siebel Installation Guide* for the operating system you are using.
- Siebel Report Viewer is automatically installed with a Siebel application when loading the Mobile Web Client environment.

- Siebel Reports Server allows running reports in multiple languages using the same instance of the Reports Server. Unlike in previous versions, there is no need to deploy a language-specific Reports Server on separate machines.

You should also be aware of the following configuration file parameters in tools.cfg (the configuration file for Siebel Tools):

- **ActuateDevWBDir.** Specifies the location of the Actuate e.Report Designer Professional software.
- **TemplateDestDir.** Specifies the directory location where data supply library files are created when they are exported from Siebel Tools. The default is C:\program files\siebel\7.7\tools\RTPSRC\ENU\LIB.

  If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \enu folder. Instead, you have a folder in the appropriate language code for your installation, such as \deu for Germany.

  See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

The following parameter in the configuration file for each Siebel application should be set as indicated:

- **EnableOLEAutomation.** By default, this is set to TRUE. This setting allows local generation of Siebel reports.

  NOTE: This is the case even if you do not have or use the Siebel Object Interfaces feature.

Setting Internal Basic Source Encoding

After Actuate e.Report Designer is installed, the setting for Internal Basic source encoding must be set to Unicode (UCS-2LE).

To make sure Internal Basic source encoding is set properly

1. From the Start menu, open Actuate e.Report Designer.
2. From the menu, choose View > Options and click General.
3. In the Internal Basic source encoding section, make sure that Unicode (UCS-2LE) is selected.

Migration Instructions for the Siebel Reports Server

To migrate the Siebel Reports Server to release 7.7, you will need to uninstall any previous version of the Siebel Reports Server. You will install the Siebel Reports Server as a new installation, following the information in the *Installing the Siebel Reports Server* chapter in the *Installation Guide* for the operating system you are using.

Migrating the Actuate Report Encyclopedia Volume

The Actuate utility, acupgrade, can be used for upgrading a preRelease 6 e.Reporting Server Report Encyclopedia volume from an earlier Actuate version to the current version. Table 5 contains a listing of releases for Siebel applications with the appropriate Actuate version used with it. For more information, see the chapter on working with iServer utilities in the *Administering Actuate iServer System* in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

Table 5. Siebel Applications and Actuate e.Reporting System Releases

Siebel Application Release	Actuate e.Reporting System Release
99 (5.x)	3.2
2000 (6.0)	4.0
6.2, 6.2.1, and 6.3	4.1
7.0	5.0
7.0 (IBM release only)	5.0 SP 2 Patch 1
7.5.2	6.0 Fix 3
7.5.2.7 (HP release only) and 7.5.2.2xx	6.0 SP1
7.5.3	6.0 SP1 Fix 10
7.7.x	7 SP2

The acupgrade utility can be used for encyclopedia volumes before Actuate 5 that need to be migrated to Actuate 7. For migrating encyclopedia volumes from Actuate 6 to Actuate 7, perform the following instructions.

To migrate Actuate 6 encyclopedia volume to Actuate 7 encyclopedia

1. Export the Actuate 6 encyclopedia volume.
2. Install the Siebel Reports Server using the instructions in the *Installing the Siebel Reports Server* chapter in the *Installation Guide* for the operating system you are using.
3. Import the Actuate 6 encyclopedia volume.

For more information on upgrading the Actuate iServer System, see the *Installing the Actuate iServer System* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

Migrating PreSiebel 6 Custom Reports to the Siebel Reports Server

CAUTION: The information in this section only applies to migrating custom reports from Siebel 99. For more information about migrating custom reports for version 6 and later, see “Migrating Custom Reports” on page 70.

Custom reports that were written for Siebel eBusiness Applications versions earlier than Siebel 6.0 will not run successfully on the Reports Server. The Actuate Basic code for methods and custom reports must be modified to achieve compatibility with the Reports Server environment. This section explains the changes that must be made to the code.

Reporting in the preSiebel 6.0 environment used OLE automation directly between a Siebel application and Actuate software, which was located on the same client machine. This kind of communication is not available for the Reports Server and Siebel Web clients.

Instead, library methods are provided that allow access to objects in a platform-independent fashion. Object variables in your custom design and library files must be replaced with integer variables that refer to the corresponding objects, and object methods must be replaced with stand-alone methods.

When a data supply library file is generated from Siebel Tools version 6.0 and later, the generated file contains updated data access logic that uses the new library methods. If you have custom reports, you must regenerate their data supply library files from Siebel Tools 6.0 and later, or they will not work in server reporting or thin client modes.

About sssiebel.bas Library Functions

Communication between Siebel and Actuate software is enabled in Siebel 6.0 and later by library functions in an Actuate Basic file, sssiebel.bas. The sssiebel.bas file has been loaded into the sssiebel.rol library in Siebel 6.0 and later, and does not need to be explicitly included in each design file. The sssiebel.bas functions provide the report with a system-independent interface to communicate with the Siebel Application Server. While sssiebel.bas functions were used in a limited set of roles in preSiebel 6.0 reporting, the role of this library file has expanded to include object interface functions.

The object interface functions in sssiebel.bas are designed to be the sole interface to the Siebel Application Server from all reports, whether standard or custom. In standard reports, the appropriate changes have already been made, and these upgraded files are installed when you upgrade the Siebel application. Your custom reports will, in most cases, run successfully on the Reports Server following regeneration of the data supply library file from Siebel Tools and recompilation of the report design in Actuate e.Report Designer Professional.

If a custom report works successfully in client reporting mode and does not work in server reporting mode, you will need to locate and replace any Microsoft Windows–specific and object interface function calls with calls to sssiebel.bas object interface functions.

Changing Object Variables to Integer Variables

The sssiebel.bas functions abstract the Siebel Application Server in terms of four basic objects: model, business object, business component, and property set. These objects can be accessed only using the sssiebel.bas functions. The report uses references of type Integer to these objects.

NOTE: Model refers to the handle to the top-level object exposed by the interface. In the case of OLE/COM, this is SiebelApplicationServer. The PropertySet object is described in the *Siebel Object Interfaces Reference*.

All variables of type Object should be changed to type Integer. Functions in sssiebel.bas take these integer references as arguments, using the references to access objects and perform operations on them.

The integer value is a handle to an automation or C++ object. Variables such as ssAppServer, ssBO, theBC, and AppServer change from object variables to integer variables. For example,

```
Set theBC = ssManaged_Person_Forecast

.

.

ssManaged_Person_Forecast.InvokeMethod(...)
```

changes to

```
theBC = ssManaged_Person_Forecast

.

.

ssBusCompInvokeMethod (ssManaged_Person_Forecast, ...)
```

Since all these variables have been converted to type Integer, they can no longer be used as objects with their own methods. Instead, special methods in sssiebel.bas are used to manipulate the objects, such as ssBusCompInvokeMethod in the preceding example. These substitute methods are described in "About Method Name Prefixing" on page 30.

NOTE: Since the variables are integers, comparison expressions can no longer use "is Nothing" and need to be changed to "= 0".

About Method Name Prefixing

Methods on business components, fields, business objects, and application servers can no longer be affixed to object variables and separated by a period. These object variables are now integer reference variables, as previously described. New stand-alone methods provided in sssiebel.bas perform the same functions previously performed by object methods. These have the same names as in the Siebel Object Interfaces, but are prefixed with ssBusComp, ssBusObj, or ssModel.

Using ssBusComp as the Prefix for Business Component Variables

Whenever an API method makes use of business component variables or names, a substitute method with a prefix of ssBusComp is used. For example, FirstRecord becomes ssBusCompFirstRecord, SuppressNotification becomes ssBusCompSuppressNotification, and so on. For example,

```
theBC.FirstRecord (errCode)
```

changes to

```
ssBusCompFirstRecord (theBC)
```

Methods with an argument list lose the errCode argument at the end of the list and acquire the business component integer argument at the beginning. For example,

```
theBC.GetFieldValue ("City", errCode)
```

changes to

```
    ssBusCompGetFieldValue (theBC, "City")
```

A reference to a specific business component, such as a business component variable, is moved into the argument list as the first argument, as follows:

```
    ssOpportunity.SuppressNotification (errCode)
```

changes to

```
    ssBusCompSuppressNotification (ssOpportunity)
```

Field methods adopt the ssBusComp prefix as well. A first argument is added for the field name, and the errCode argument is dropped. For example,

```
    ssAccount.SetSearchExpr (searchSpec, errCode)
```

changes to

```
    ssBusCompSetSearchExpr (ssAccount, searchSpec)
```

and

```
    ssAccount.SetViewMode (ssReport::ssViewMode, errCode)
```

changes to

```
    ssBusCompSetViewMode (ssAccount, CInt (ssReport::ssViewMode))
```

This latter example illustrates the setting of the view mode (visibility). The view mode can be explicitly set to user view, manager view, and so on. For more information, refer to SetViewMode in the *Siebel Object Interfaces Reference*. The ssBusCompSetViewMode function expects the ssViewMode value in integer format, but the parameter in ssReport is a string and needs to be converted.

Using ssModel as the Prefix for AppServer-Related Methods

AppServer-related methods are prefixed by ssModel. The errCode argument is removed, and the application server name or variable becomes the new first argument. For example,

```
    ssAppServer.ActiveBusObject (errCode)
```

changes to

```
    ssModelActiveBusObject (ssAppServer)
```

and

```
    ssAppServer.GetBusObject (ssReport::ssBusObjectName, errCode)
```

changes to

```
    ssModelGetBusObject (ssAppServer, ssReport::ssBusObjectName)
```

Using ssBusObj as the Prefix for Business Objects

The names of methods related to business objects are prefixed by ssBusObj. The errCode argument is removed, and the business object name or variable becomes the new first argument. For example,

```
busObjName = ssBO.Name (errCode)
```

changes to

```
busObjName = ssBusObjGetName (ssBO)
```

and

```
ssBO.GetBusComp ("Account", errCode)
```

changes to

```
ssBusObjGetBusComp (ssBO, "Account")
```

Replacing InvokeMethod for Use with Business Component

InvokeMethod is replaced by ssBusCompInvokeMethod when it is used by a business component. It is replaced by ssModelInvokeMethod when it is used by appServer or ssAppServer. Also, in both cases, the errCode parameter is no longer used.

For an invoked method on a business component,

```
theBC.InvokeMethod ("Account Status", errCode)
```

changes to

```
ssBusCompInvokeMethod (theBC, "Account Status")
```

For an invoked method on the application server,

```
AppServer.InvokeMethod (parameters)
```

changes to

```
ssModelInvokeMethod (parameters)
```

About Message Boxes

Reports often use message boxes for reporting errors. However, message boxes are relevant only on the Windows platform on a client machine, and do not work when reports are being generated in batch mode on a Reports Server. sssiebel.bas provides operating-system-independent functions for error reporting, with behavior appropriate to the environment in which the report is executed.

The MsgBox function in method code in ROL and ROD files is replaced with ssDisplayMessage or ssProcessLastError, depending on whether an AppServer variable is used. For a message box that presents an information string, ssDisplayMessage is used, as follows:

```
MsgBox (string)
```

changes to

```
ssDisplayMessage (string)
```

A message box that makes use of the AppServer or ssAppServer variable uses the ssProcessLastError function.

```
MsgBox ssAppServer.GetLastErrText ()
```

changes to

```
    ssProcessLastError (ssAppServer, "", "")
```

The syntax for the ssProcessLastError function is as follows:

```
    ssProcessLastError (object_integer, pre_message_string, post_message_string)
```

Changing CreateObject to ssConnect

CreateObject changes to ssConnect. ssiSiebelServer is a global integer variable that keeps track of the connection. For example,

```
If (ssAppServer is Nothing) Then

Set ssAppServer = CreateObject (ssReport::ssOLEServer)

If (ssAppServer is Nothing) Then

Start = False

Exit Function

End If

End If
```

changes to

```
ssReport::ssiSiebelServer = ssConnect(ssReport::ssOLEServer)

If (ssReport::ssiSiebelServer = 0) Or (errCode <> 0) Then

ssDisplayMessage ("Login Failed!")

Exit Sub

End If
```

3 Reporting in the Siebel Web Clients

This chapter describes the user interaction with the Reports Server product in the Web Client and Dedicated Web Client (both connected and disconnected modes).

Siebel Reports Server is an out-of-the box integration of the Actuate iServer System. While the end user interaction with Siebel Reports Server will be completely within the Siebel application, the administrator still needs to administer the reports encyclopedia using Actuate Management Console. In order for the users to run reports in the Web Client and connected Dedicated Web Client, the administrator should have installed the Siebel Reports Server as described in the *Installation Guide* for the operating system you are using.

NOTE: In the disconnected mode of the Dedicated Web Client, reports can only be run interactively and the Reports Server views will not be available.

The following topics are included in this chapter:

- "About Reporting in the Web Client" on page 35
- "About Reporting in the Dedicated Web Client" on page 40
- "Changing the Locale and Language for Reports" on page 43

About Reporting in the Web Client

Siebel Reports Server allows Web Client users of Siebel eBusiness Applications to run reports both in interactive and scheduled modes. To run a report in a view, click the Reports button on the toolbar. The resulting drop-down listing allows the user to select a report available for that view. From the Reports button, users can also use the Schedule Report option to run the report in scheduled mode at a later time. The Schedule Report option displays a separate dialog for the user to enter the schedule parameters.

About the System Architecture for Reporting in the Web Client

The Reports Server encyclopedia consists of report executables and the user folders. These report executables correspond to the complete set of available reports among all the views. These executables run in the Reports Server factory process (a multi-threaded report execution process) at report execution time and generate the report output file (ROI) by obtaining data from the Siebel Object Manager. The report output (ROI) file will be stored in the user folder in the reports encyclopedia and can be accessed from the Reports Server views in Siebel eBusiness Applications. The report is executed by passing the parameter (ROV) file, which is generated by Siebel Object Manager when the user runs a report from a Siebel eBusiness Application view.

Figure 4 illustrates the report execution process from the Web Client.

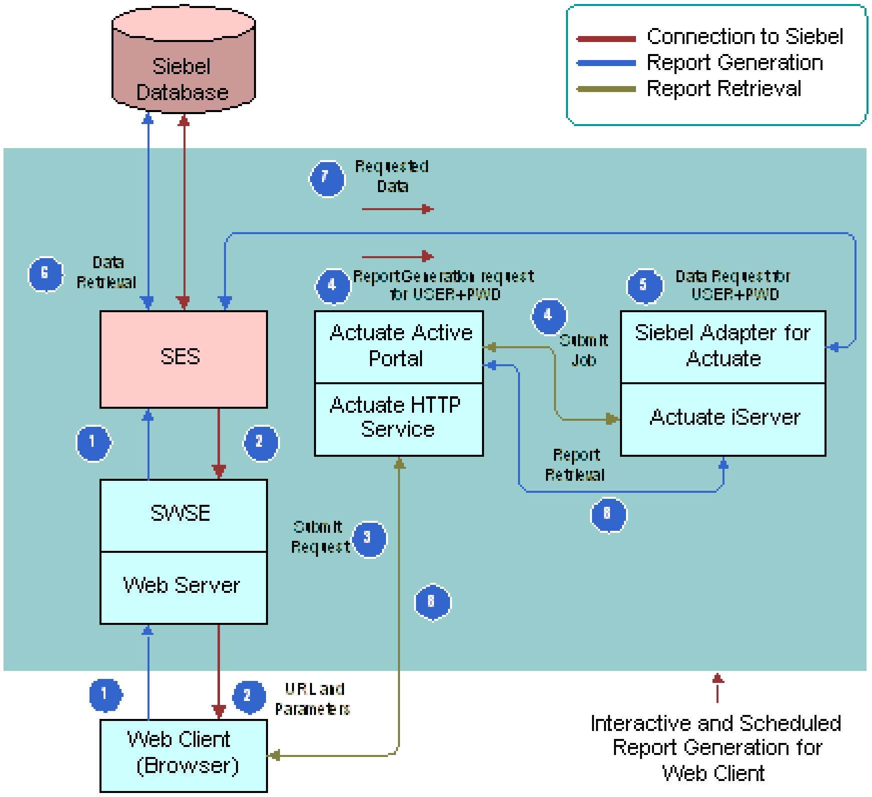


Figure 4. Server-Based Reporting for the Siebel Web Client

The following is the sequence of report generation steps, corresponding to Figure 4:

1. The report request is submitted by the browser through Siebel Web Services Extension (SWSE) to the Siebel Server.
2. The Siebel Object Manager (OM) creates and passes a URL for Active Portal with the Siebel Report Server login parameters, and the parameters to generate the report, to the browser.
3. The browser submits the request to Active Portal.
4. Active Portal authenticates the user on the Reports Server and requests the Reports Server to generate the report.
5. The Reports Server runs the report and requests data from the OM through the Siebel Adapter for Actuate.
6. Data is retrieved by the Siebel OM from the Siebel database and provided to the Reports Server.
7. The report initiates using progressive report generation and retrieval. As soon as the first page is ready, the Reports Server creates the DHTML and starts delivering the report to the browser through Active Portal.

8 The complete report in DHTML is displayed in the Web Client.

The report request submitted from the Web Client includes parameters for the current view, active query, sort specifications, and visibility rules. For all reports run in the batch mode (using the Schedule Report option), the status of submitted report requests can be obtained from the Reports Server views.

NOTE: The user need not be logged into a Siebel eBusiness Application for the scheduled reports to be generated, since report generation by the Reports Server is asynchronous. The data for report generation is obtained from the Siebel Object Manager running under the Siebel Server.

Requesting Reports in the Siebel Web Client

Users can request reports by using the Reports button that is located on the toolbar. The next section explains how to request a report using the Reports button.

To request a report in the Siebel Web Client

1 Click the Reports button.

2 Select the report to generate from the report listing in the drop-down menu.

From the drop-down menu of the Reports button, you can review your previously requested reports by selecting My Reports. For more information, see "About the My Jobs View" on page 38.

Using the SOAP protocol, reports submitted from the Siebel Web Clients are run on the exports Server as a transient job. Reports are no longer run as a background job. In this run now mode, the report is viewed as it is being generated. Using progressive viewing, the first page is displayed before the entire report is completed.

However, if the report generation fails, an error message will not be available as no completed notification is generated. Use the scheduling function to request the report again in order to review what is causing the report to fail.

Scheduling Reports in the Siebel Web Client

Scheduling reports is also possible by selecting the Schedule Report from the drop-down menu after clicking the Reports button. The Schedule Report window that appears includes a drop-down listing that allows you to select the report you are scheduling.

The Schedule a Report window allows you to specify the time and date when the report is run, select the option to generate the report with a recurring frequency, and indicate if the report should be printed upon generation.

To schedule a report in the Siebel Web Client

1 Click the Reports button and select Schedule Report.

2 On the *Schedule a Report* window, select the report to schedule from the drop-down list.

   Click Schedule.

   You can schedule a report for one-time or periodic generation.

3 In the *Schedule this job* section of the Schedule tab, click Once and set the date and time parameters for your report request.

   You can also request to schedule the report right now or to set it as a recurring report. If set as recurring, select how often you want the report to run and enter the parameter values for the starting and ending dates.

   Click the Output tab.

4 (Optional) Accept the default or change the document and version names.

5 (Optional) If the report file with the same name already exists, select either *Create a new version* or *Replace the latest version*.

   You can also specify the number of versions of the report that you would like to keep on the Reports Server.

   Click Submit.

You can view the status of the report from the Action status window. Click Close to close the window and return to the original view in the Siebel application.

NOTE: Selecting *Cancel job* on the Action status window will delete the scheduled report request.

After the report is scheduled, you can review the My Jobs view to also check the status of the job.

About the My Jobs View

This replaces the Active Requests, Scheduled Requests, and Completed Requests Notifications views. My Jobs can be accessed to view the status of reports that you have submitted using Schedule Report. Access My Jobs by navigating to Site Map > Reports Server > My Jobs.

The four tabs in My Jobs: Scheduled, Pending, Running, and Completed, display reports that are being processed:

- Schedules reflects the jobs that will be processed at a later date and time.
- Pending reflects the jobs that are currently scheduled for processing.
- Running reflects the scheduled jobs that are currently being run.
- Completed shows the scheduled jobs that are processed.

After the job completes, you can review the My Reports view, which is a collection of reports that you have generated.

About the My Reports View

The My Reports view displays the report output files that you have access to in the Reports list. This includes the reports generated by you, and the reports that other users granted you permission to view. The view is actually being generated directly from Actuate Active Portal.

By clicking the reports folder, you will see a listing of the reports that have been requested. Clicking the name of the report will produce another copy without you having to request the report again through the screen views.

The Reports Administrator can also provide access to the Personal Profile view. Using this view, end users can set an email address, default printer, notification preferences, and other functions as needed.

About DHTML Report Viewer Keyboard Shortcuts

The generated report is viewed in the DHTML report viewer for the Web Client. For this viewer, the user interface functions are available as keyboard shortcuts. Table 6 lists the mapping of keyboard shortcuts to the buttons in the DHTML report viewer.

Table 6. DHTML Report Viewer Keyboard Shortcuts

Button	Keyboard Shortcut
Save	S
Print	P
Table of Contents	C
Search	E
Go	G
First Page	F
Previous Page	B
Next Page	N
Last Page	L

About Reporting in the Dedicated Web Client

Reports can be generated both in the connected and the disconnected mode of the Dedicated Web Client. In the disconnected mode of the Dedicated Web Client, reports can only be generated interactively by accessing data from the local Object Manager. Further, interactively run reports (both in connected and disconnected mode) are displayed in the ActiveX report viewer (not in the DHTML report viewer). Finally, reports can be scheduled only in the connected mode of the Dedicated Web Client. Scheduled reports run on the Reports Server and use the Siebel Object Manager

About System Architecture for Reporting in the Dedicated Web Client

When a report is generated in the Siebel Web client, the Reports Server encyclopedia volume and the Client consist of the report executables. These report executables correspond to the available reports among all the views.

When a report is run interactively in the Dedicated Web Client (connected or disconnected mode), the corresponding report executable is started in the Client. The data for the report is obtained from the local Object Manager, and the report is displayed in the ActiveX viewer in the Client.

In the connected mode of Dedicated Web Client, users can submit report requests to be run in schedule mode in the Reports Server. These requests are submitted to the Reports Server in the form of a parameter (ROV) file. At report execution time, the report executable obtains data from the Siebel Object Manager and generates the report output file (ROI). The report output (ROI) file will be stored in the user folder in the reports encyclopedia and can be accessed from the Reports Server views in Siebel eBusiness applications.

Figure 5 illustrates the report execution in the Dedicated Web Client.

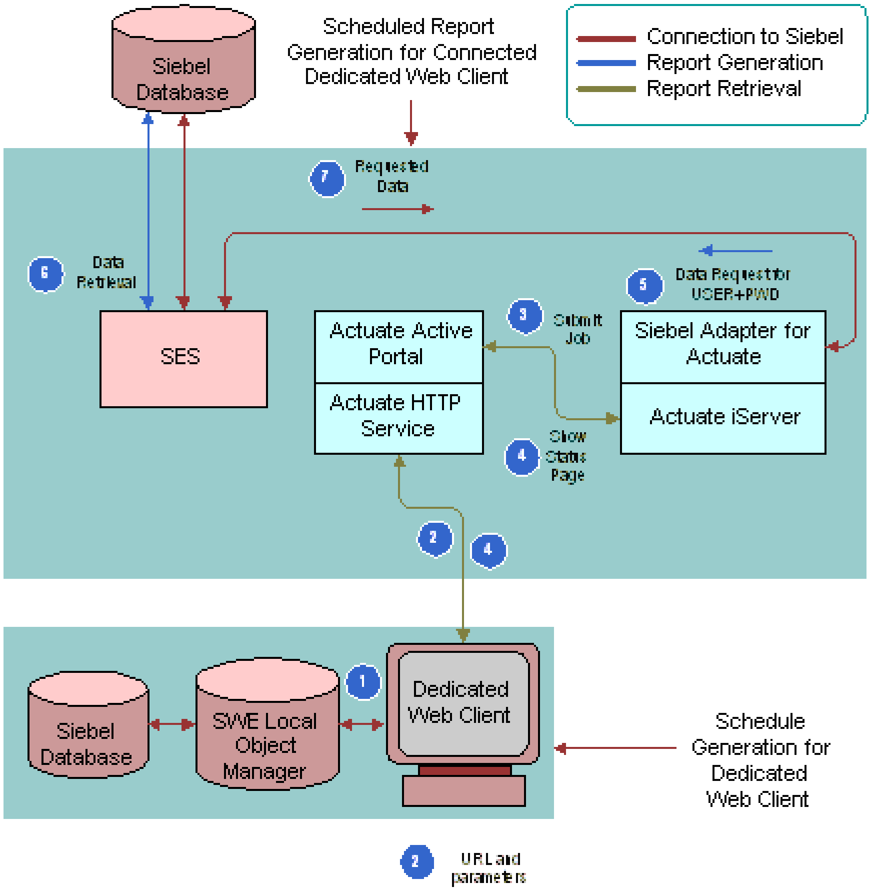


Figure 5. Server-Based Reporting for the Siebel Dedicated Web Client

The following is the sequence of report generation steps, corresponding to Figure 5, in schedule mode in the connected Dedicated Web Client:

1 The report request is submitted by the browser to the Local Object Manager (OM).

2 The Local OM creates and passes a URL for Active Portal with the Siebel Reports Server login parameters to the browser.

3 The browser submits the request to Active Portal.

4 The Dedicated Web Client connects to Active Portal, which authenticates the user on the Reports Server and requests that the report be generated.

5 Once connected, the Reports Server requests data from the OM through the Siebel Adapter for Actuate.

6 Data is retrieved by the Siebel OM from the Siebel database.

7 The requested data is provided to the Siebel Adapter for Actuate.

NOTE: The user need not be logged into a Siebel eBusiness application for the scheduled reports to be generated, since report generation by the Reports Server is asynchronous and data for report generation is obtained from the Siebel Object Manager.

Requesting Reports in the Dedicated Web Client

The user interface and process flow section here is similar to the information in "About Reporting in the Web Client" on page 35. The following information reflect any changes that are specific to the Dedicated Web Client.

Connected Mode

Reports can be generated in the Dedicated Web Client in the connected mode. However, reports run locally obtained data from the local database and not from the Reports Server when in connected mode. Reports can be scheduled and run from the Reports Server.

Disconnected Mode

Reports can be generated in the Dedicated Web Client in the disconnected mode also. However, the data for report generation is obtained from the local object manager. The report generation process in the disconnected mode differs from the connected mode as described below:

- Reports are run locally by obtaining data from the local database.
- Reports cannot be scheduled.

About ActiveX Report Viewer Keyboard Shortcuts

The generated report is viewed in the ActiveX report viewer (the Siebel Reports Viewer) for the Dedicated Web Client. For this viewer, the user interface functions are available as keyboard shortcuts. Table 7 lists the mapping of keyboard shortcuts to the buttons in the ActiveX report viewer.

Table 7. ActiveX Report Viewer Keyboard Shortcuts

Button	Keyboard Shortcut
Open	O
Save As	S
Print	P
Print Setup	U
Send Report	R
Table of Contents	C

Table 7. ActiveX Report Viewer Keyboard Shortcuts

Button	Keyboard Shortcut
Search	E
First Page	F
Previous Page	B
Next Page	N
Last Page	L
Go To Page	G
Stop Report In Progress	Q

About Searching in the Siebel Report Viewer

In the DHTML report viewer, you can search in a report and export the results to a Comma Separated Value or Tab Delimited Value format.This functionality is added to the ActiveX report viewer in the 7.5 release. The new Search button is located next to the Table of Contents button. Essentially, the user interaction is the same as that in the DHTML report viewer.

For more information, see the chapter on searching in Actuate reports in the *Viewing Actuate Basic Report* manual located in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

Changing the Locale and Language for Reports

If you need to designate a language and locale that is different from the one your Siebel application is currently using, then you will change the report parameters in User Preferences.

NOTE: The corresponding language pack must be installed on the Siebel Reports Server in order to use this feature.

To change the locale and language for reports

1. From the application-level menu, select Tools > User Preferences.
2. Select Report Parameters from the link bar.

   The Behavior form applet appears.
3. Click the Locale select button and select a locale from those available.

   The locale selected will designate the locale setting to use for formatting and displaying the report data.

   Click OK to save the locale.

4 Click the Language select button and select a language from those available.

The language selected will designate the language of the static text that will be displayed in the report.

Click OK to save the language.

The language and locale of the report are generated independently of the locale in which the user is currently running Siebel eBusiness Applications. The language and locale of reports will remain as selected until changed on the Reports Parameter view.

If no strings are found for the language and locale, labels and properties are defaulted to the original ROD file values for that report.

About the User Administration View

The User Administration view is only available to an administrator. This view allows the administrator to automatically create and synchronize accounts for Siebel report users in the Actuate encyclopedia volume. The administrator can synchronize one account or multiple accounts at once from this view on an ongoing basis. It is recommended that the administrator run a query in this view to synchronize users in smaller buckets, particularly when a large number of users should be synchronized.

As the result of synchronization, an account with the same user ID that appears in the Siebel applications is created on the Reports Server. Also, a default folder with this name is created on the Reports Server, and basic privileges such as read, write, and execute are granted to that user to access the folder. Each Reports Server account is created with the same user ID as the Siebel user ID and a password. This user ID and password are stored in an encrypted format in the Siebel database, to be passed to the Reports Server when needed.

The navigation path to this view is Navigate > Site Map > Reports Server > User Administration. An administrator account is established in the Actuate encyclopedia as part of the postinstallation tasks of the Reports Server (see the *Siebel Installation Guide* for the operating system you are using). This account allows the administrator to log in to the Actuate Management Console to perform ongoing administration tasks such as administering and cleaning up the encyclopedia volume. For more information, see "About Maintaining the Siebel Reports Server" on page 45.

After performing the *Synchronize One* or *Synchronize All* operations, accessible from the User Administration view, review the Application Object Manager log file (such as the Callcenter OM log) for errors that might have occurred, for example:

```
SyncAll : Error occurred during user creation for user X, errcode = Y
```

```
SyncOne : Error occurred during user creation for user X, errcode=Y
```

Synchronization errors are not reflected in the user interface.

4 Managing the Siebel Reports Server

For better performance, the reports administrator can perform certain functions on the Siebel Reports Server.

The topics included in this chapter are:

About Maintaining the Siebel Reports Server

The administrator should perform certain ongoing maintenance tasks to assure the continued performance of the Actuate iServer System. In addition to maintaining the Reports Server performance, the administrator should perform user administration, which includes setting up users and roles, creating and modifying their preferences, configuring printers, creating auto-archive policies, and maintaining security of the encyclopedia. For details of administration tasks, refer to the *Administering Actuate iServer System* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

About Managing Process Groups

The administrator may create new Factory/View/Print Process Groups as part of managing the Reports Server resources. The administrator may increase the number of factory processes to have multiple reports generated simultaneously. For more information on managing process groups, see the *Administering Actuate iServer System* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

About Port Configuration

The administrator can open more free ports to the Reports Server (particularly in a firewall environment) to enhance the number of concurrent user connections for report generation and viewing. Actuate's *Administering Actuate iServer System* describes port configuration both for report generation and viewing.

About Periodic Shutdown

The Actuate iServer System maintains a lookup table of the contents of the encyclopedia volume and updates this table when the Reports Server is shut down and restarted. The lookup table size increases with the number of completed request notifications, report files (ROI), users, or roles. The lookup table does not always automatically decrease in size when these are deleted unless the report server is restarted. Therefore, the administrator should periodically shut down and restart Actuate services (in Windows) or processes (in AIX and Solaris).

Increasing the Maximum Number of Factory Processes

Increasing the maximum number of factory processes allows for increased scalability when using the Reports Server. The number to use is dependent on the number of CPUs that are available to the Reports Server. Maximum factories for synchronous and asynchronous jobs should be set based on your business requirements and hardware availability. This is applicable for all platforms.

To increase the maximum number of factory processes

1. From the Start menu, navigate to Actuate Management Console and select System Administration to log into. Click Log In.
2. Click on System Resource Groups and select *Default Sync* for synchronous jobs or *Default Async* for asynchronous jobs.
3. Click Server Assignments.
4. Change the Max Factories field as required by the business process and click OK.
5. Log out of System Administration.

5 Global Report Modifications

Global customization applies to all reports, rather than to a single report. It is accomplished by making changes to components in the sscustom library and propagating those changes to all the reports that use those components.

This chapter consists of the following topics:

About Global Report Customization

Global layout customization provides a good introduction to Siebel report customization, because it is relatively quick and straightforward and generally involves modifications to a small number of property settings. Typical global modifications are those made to the header or footer, such as putting a different company name or logo in the header. Another global modification might be setting a different default font size and style for all reports.

A change to a library component is a global report modification because it affects all the reports that use that component. You always make global modifications in the sscustom library. For example, if you edit the font property of the ssLbl (label) component in the sscustom library, any part of a report that uses ssLbl will use the new font that you specified. You must recompile each report for the change to affect that report.

The procedures in this chapter demonstrate how to modify the page layout objects in the sscustom library, then recompile one report so that the global layout change is applied in that report. Two example procedures are provided:

- Changing the font that is used on all the text and label controls throughout the reports
- Changing the company logo that is used on all the reports

NOTE: After a global report modification, recompiling reports in batches is possible using the Actuate e.Reporting Designer Professional executable, Erdpro.exe. For more information, see the section on upgrading reports to Actuate 7 in the *Upgrading to Actuate iServer System Suite 7* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

Changing the Font on All Reports

You can change the font that is used on all the text and label controls throughout the reports. You change the Font.FaceName property on the ssTxt (text) library control and on the ssLbl (label) library control, as shown in the following steps. All report text and label controls are derived from ssTxt and ssLbl.

In order to make and test the appropriate changes, you will need to:

- Open a report design file.
- Edit the label and text controls in sscustom.
- Compile and run the report.
- Save the changes to sscustom.

To open a report design file and the corresponding Siebel view

1. Open Siebel Sales.
2. From the application-level menu, choose View > Site Map > Accounts > My Accounts.

   This step is necessary so that Siebel data can be communicated to the Actuate report when the report is tested.
3. Open the Actuate e.Report Designer Professional software from the Start menu.
4. Choose File > Open, then locate and select Aclist.rod in the \Siebdev\\RPTSRC\\ENU\\STANDARD folder.

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \\enu folder. Instead, you have a folder in the appropriate language code for your installation, such as \\deu for Germany.

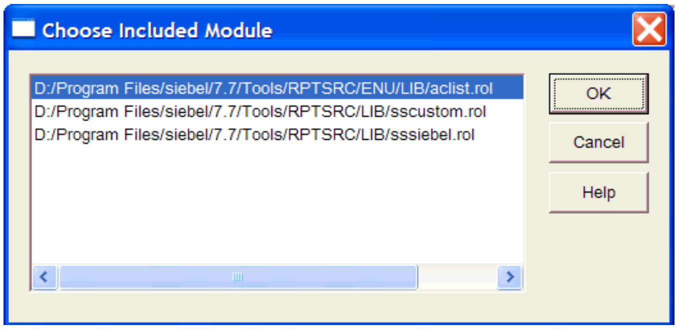
   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

   The Design Editor window displays the Account List report design.

To edit the label and text controls in sscustom

1. Click the Library Browser button.

   The Choose Included Module dialog box appears as shown in the following figure.



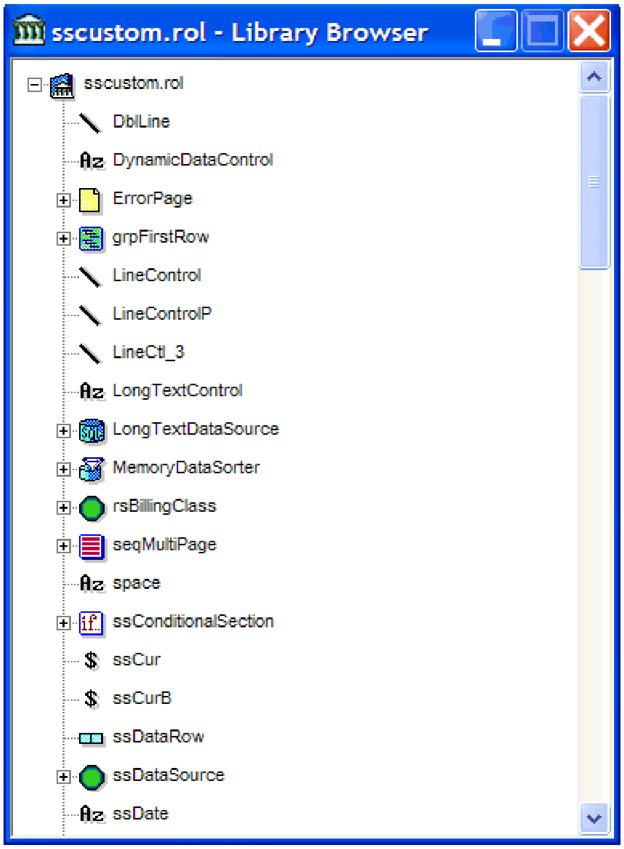
D:/Program Files/siebel/7.7/Tools/RPTSRC/ENU/LIB/aclist.rol
D:/Program Files/siebel/7.7/Tools/RPTSRC/LIB/sscustom.rol
D:/Program Files/siebel/7.7/Tools/RPTSRC/LIB/sssiebel.rol

The Library Browser button and the Choose Included Module dialog box are for opening one of the libraries that has been included in the report design. For this example, the library for Account List (aclist.rol) is included. The sscustom and sssiebel libraries are included in all reports.

2 Select sscustom.rol and click OK.

As shown in the following figure, the Library Browser window for sscustom.rol appears.

The Library Browser window displays all of the library components available in the library you opened, in this case sscustom.rol. This library contains such components as ssLbl and ssTxt that are modified in the following steps.

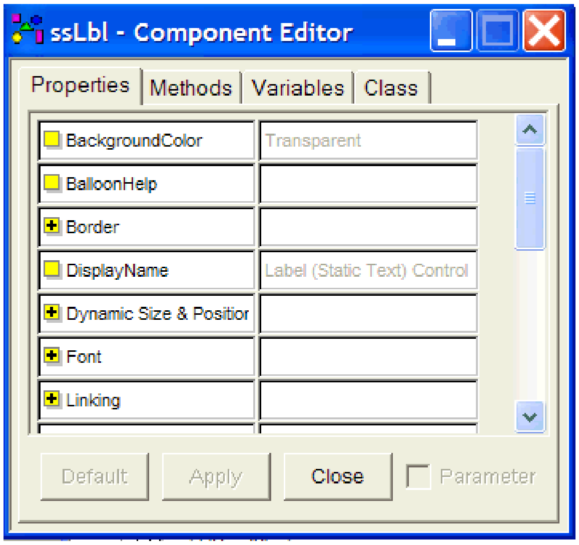


sscustom.rol - Library Browser

- sscustom.rol
  - DblLine
  - DynamicDataControl
  - ErrorPage
  - grpFirstRow
  - LineControl
  - LineControlP
  - LineCtl_3
  - LongTextControl
  - LongTextDataSource
  - MemoryDataSorter
  - rsBillingClass
  - seqMultiPage
  - space
  - ssConditionalSection
  - ssCur
  - ssCurB
  - ssDataRow
  - ssDataSource
  - ssDate

3 Right-click the ssLbl component and select Properties.

The Component Editor window appears, displaying the property settings for ssLbl as seen in the following figure.



BackgroundColor	Transparent
BalloonHelp	
Border	
DisplayName	Label (Static Text) Control
Dynamic Size & Positior	
Font	
Linking	

The Component Editor window has four tabs—Properties, Methods, Variables, and Class:

- **Properties tab.** Identifies the property setting for each property defined for the current component. Property settings can be changed in this tabbed page.
- **Methods tab.** Lists the Actuate BASIC methods defined for the component.
- **Variables tab.** Lists the variables defined for the component.
- **Class tab.** Identifies the class name and superclass of the component, and the module where it resides (usually either the report design file or an included library).

4 Find and expand the Font property in the Property Editor.

The Font.FaceName property becomes visible.

5 Change the value of Font.FaceName from Arial to Arial Narrow and click Apply.

6 Click the Library Browser window to make it active.

You do this to choose a different library component to change its properties.

7 Click the ssTxt component in the Library Browser window.

The Component Editor window now shows ssTxt property settings.

8 Repeat Step 4 and Step 5 on page 50 for ssTxt and close the Component Editor and Library Browser windows.

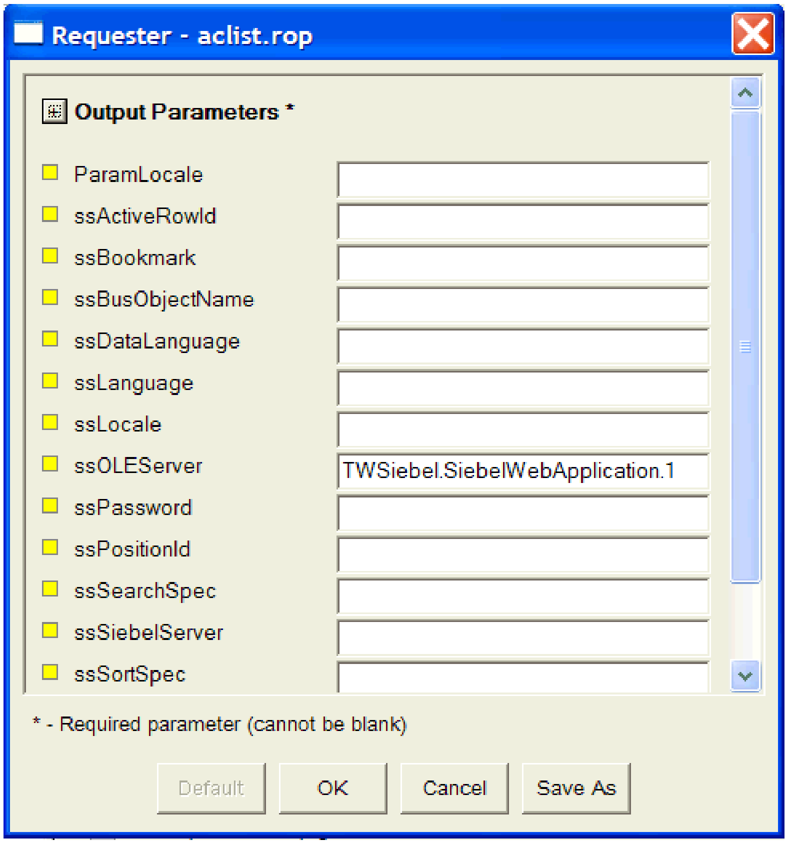
To compile and run the report

1 In Actuate e.Report Designer Professional, click the Run button.

The Run button invokes the report build, compilation, and execution processes. You can also invoke these processes, collectively or individually, from the Report menu in Actuate e.Report Designer Professional.

2 The Actuate e.Report Designer Professional Requester dialog box appears.

As the following figure represents, the Requester dialog box appears containing a listing of the parameters for the report such as ssOLEServer and ssPassword.



ParamLocale	
ssActiveRowId	
ssBookmark	
ssBusObjectName	
ssDataLanguage	
ssLanguage	
ssLocale	
ssOLEServer	TWSiebel.SiebelWebApplication.1
ssPassword	
ssPositionId	
ssSearchSpec	
ssSiebelServer	
ssSortSpec	

* - Required parameter (cannot be blank)

Click OK to accept any defaults. This provides the means to communicate parameter values to the report.

The report is generated and appears in the Actuate e.Report Designer Professional window.

To save the changes to sscustom

1 Close the report.

2. If the Save Modified Modules dialog box appears, this is to notify you that sscustom has been modified and to ask if you want to save the changes.

   Click Yes.

3. Close Actuate e.Report Designer Professional.
4. Copy the Aclist.rox file from the development directory (\Siebdev\RPTSRC\language_code\STANDARD) to the executables directory (\Siebel_client\REPORTS\language_code).
5. In Siebel Sales, select Account List from the Reports menu.

   The revised Account List report appears in the browser screen.

For your change to sscustom to affect other reports, you will need to recompile those reports.

Changing the Corporate Logo on All Reports

You can change the company logo that is used in all the reports. You change the fileName property of the ssLblSiebel control, as described in the procedures in this section.

To make and test the appropriate changes, you will need to:

- Open a report design file.
- Edit the ssLblsbl component in sscustom.
- Compile and run the report.
- Save the changes to sscustom.

To open a report design file and the corresponding Siebel view

1. Open Siebel Sales.
2. From the application-level menu, choose View > Site Map > Accounts > My Accounts.
3. Open the Actuate e.Report Designer Professional software from the Start menu.
4. Choose File > Open.
5. Locate and select Aclist.rod in the \Siebdev\RPTSRC\ENU\STANDARD folder.

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

   The Design Editor window displays the Account List report design.

To edit the label and text controls in sscustom

1. Click the Library Browser button.

   The Choose Included Module dialog box appears.

2. Select sscustom.rol and click OK.

   The Library Browser window for sscustom appears.

3. Locate and expand ssReport, PageList, and PageStyle.
4. Right-click the ssLblSiebel component and select Properties.

   The Component Editor appears.

5. Replace the filename specified in the fileName property.

   You can use a graphic file with your company's logo to replace the Siebel logo graphic. Recompiled reports will now display your company's logo.

   NOTE: If there is no logo bitmap for your company, choose any existing bitmap file in the \Siebdev\\RPTSRC\LANGUAGE\\STANDARD (where *LANGUAGE* stands for the Language Pack you want to deploy, such as ENU for U.S. English) directory for demonstration purposes. You will need to change the property setting back to sslogo5.bmp and recompile later.

6. Click Apply, and save the changes.
7. Click Close to close the Component Editor, and then close the Library Browser window.

To compile and run the report

1. Click the Run button to compile and run the report.
2. Click OK in the Requester dialog box to accept the defaults.

   The report is generated and appears in the Actuate e.Report Designer Professional window.

To save the changes to sscustom

1. Close the report, then close the report design.
2. Copy the Aclist.rox file from the development directory to the executables directory.
3. In Siebel Sales, click the Reports button and from the drop-down list select Account List.

   The revised Account List report appears in the browser window.

The majority of Siebel reports are produced using a landscape orientation (ssLblSiebel control from ssPage pagestyle). For reports that use a portrait orientation, you use ssLblSiebelP from ssPagePortrait. All other instructions remain the same.

6 Creating a Simple List Report

New report creation involves putting together report elements from scratch in Actuate e.Report Designer Professional. It is the most commonly employed technique for satisfying the custom reporting requirements of an organization. There are also some shortcut techniques, discussed in this chapter, to speed up the creation of a new custom report. The chapter also includes information on migrating custom reports to a new Actuate version.

The chapter contains the following topics:

- "Creating a New Report Versus Subclassing a Design" on page 55
- "How a Simple List Report Works" on page 56
- "Example—Creating a Simple List Report" on page 59
- "Alternative Report Creation Strategies" on page 68
- "Additional Information for Developing and Deploying Siebel Reports" on page 70

Creating a New Report Versus Subclassing a Design

New report creation is in contrast to report subclassing. You create a new report when your requirements are not satisfied by any existing report and there are significant differences between your desired report and any existing report. You obtain a new report by subclassing when the differences between your new report and an existing report are minor and you want the new report to include any upgrades made to the old one.

The following are some situations in which you create a new report:

- When the list of fields in the report's object definitions differs from that in the existing report by more than a small number of fields.
- When the component structure of the report differs from those of related reports for example, with the addition or removal of a group section or subreport.

  NOTE: Group sections are described in Chapter 7, "Reports with Group Sections." Subreports are described in Chapter 8, "Master-Detail Reports."

- When the custom report uses a new business component.

The following are some situations in which you subclass an existing report design:

- When you are deploying multiple similar versions of the same report showing slightly different data to different categories of users.
- When two reports show the same data but present it differently.

How a Simple List Report Works

Figure 6 illustrates the structure of a simple list report.

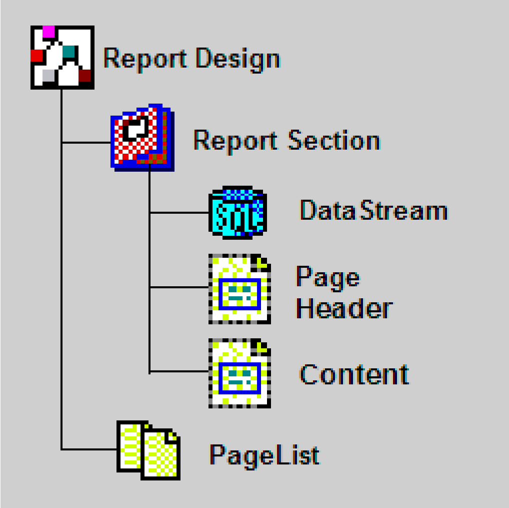


Figure 6. Simple List Report Structure

This list report includes the following major components:

- **Report Design.** This is the top-level component in a report; it corresponds to the ROD file in which it resides. In a simple report, you do not need to modify the top-level component; you only add child components to it. In a more complex design, you may add global variables to the report design component.
- **Report Section.** A report section groups together components that define the source of data, physical layout, and behavior of a master report or subreport.
- **DataStream.** A datastream component defines the source of data for a report section. In simple Siebel reports, the datastream always consists of the contents of the data supply library (ROL) file, which defines the transfer of data from a Siebel view to the Actuate report. In more complex reports, additional datastreams may manipulate the data obtained from the library datastream.
- **Page Header Frame.** A frame is a rectangular layout area for data controls, labels, and other visual components. The contents of the page header frame are displayed at the beginning of each new page. In a simple list report, the page header shows only the column headings for the report columns. In a more complex report with group breaks, the page header shows group break information, such as the account name and location for each group of opportunities by account.
- **Content Frame.** The content frame defines the layout of one report row. Each data record that is obtained through the datastream is formatted according to the layout of data controls in the content frame.

- **Pagelist Section.** The pagelist section defines the page layout of the report, including information to be presented at the top and bottom of the page, such as the report name, page number, and company logo. It also defines the area of the page that can be used for presenting data generated in report sections. You incorporate a standard Siebel report pagelist component and make modifications to a few properties, such as the report title.

Examining an Existing List Report

To learn more about the configuration of a simple list report, it is helpful to open a standard Siebel report of this type in Actuate e.Report Designer Professional and study it. A good report to study for this purpose is the Activity List report, which is invoked from a view in the Activities screen (typically the My Activities view). You should examine the report output in Siebel Sales, and the report design in Actuate e.Report Designer Professional, leaving both open to compare them.

To generate the Activity List report in Siebel Sales

1. Open Siebel Sales.
2. Click the Site Map and navigate to Activities > Activities Home > My Activities.
3. Select Activity List from the link bar.
4. Click the Reports button and from the drop-down list, select Activity List.

The Activity List report appears in the browser window.

To open the report design for the Activity List report

1. Open Actuate e.Report Designer Professional.
2. Choose File > Open from the menu bar.
3. In the Open dialog box, navigate to the \RPTSRC\STANDARD directory in your Siebel Tools installation directory; for example, \Siebdev\RPTSRC\STANDARD and choose Actlist.rod.

   NOTE: Actlist.rod is not to be confused with Aclist.rod, which is the Account List report. The correct one to open here is the Activity List report—Actlist.rod.

Figure 7 shows the Actlist.rod report design file in Actuate e.Report Designer Professional.

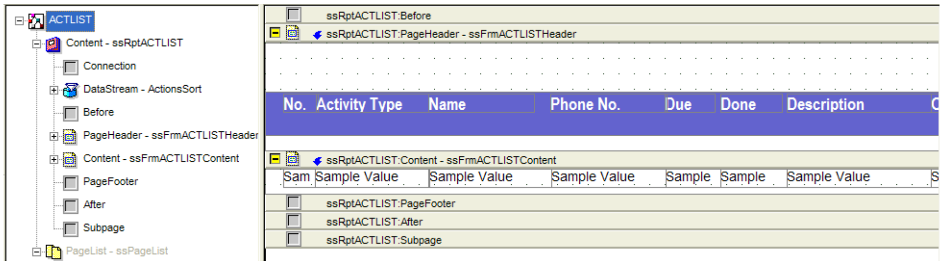

ACTLIST
Content - ssRptACTLIST
Connection
DataStream - ActionsSort
Before
PageHeader - ssFrmACTLISTHeader
Content - ssFrmACTLISTContent
PageFooter
After
Subpage
PageList - ssPageList
ssRptACTLIST:Before
ssRptACTLIST:PageHeader - ssFrmACTLISTHeader
No. Activity Type Name Phone No. Due Done Description
ssRptACTLIST:Content - ssFrmACTLISTContent
Sam Sample Value Sample Value Sample Value Sample Sample Sample Value S
ssRptACTLIST:PageFooter
ssRptACTLIST:After
ssRptACTLIST:Subpage

Figure 7. Activity List Report Design

Explore the report design, by expanding and closing folders in the tree diagram on the left, comparing design elements to the corresponding features in the report output, and right-clicking components to view their property lists.

Notice some features of this report design:

- Component names that are light gray are being referenced in a library. Component names in black have been subclassed, making them available for local modification in the report without affecting the original. The same color scheme is used for methods. The name of a method (in the Methods tab in the Properties window for a component) that has been locally modified is black. The names of unmodified methods obtained from the superclass of the component are light gray.
- The name of the datastream component, ActionsSort, is gray, indicating that it is referenced, and not local. If you open the Properties window for this component and click the Class tab, you can see that the original is in the Actlist.rol library file. Actlist.rol is the data supply library file generated from Siebel Tools.
- If you expand the datastream component, you can see that it has a child data row component, ssActionDataRow. This, too, is referenced rather than local, and comes with ssActionQuery from the data supply library file. If you open the Properties window for ssActionDataRow and click the Variables tab, you can see that the data row component consists of variables (such as ssAccount_Location, ssAccount_Name, and ssContact_First_Name) that are derived from the list of fields in the business component record supplying the data.

  NOTE: Specifically, these fields are generated from the list of report field children of the report object definition from which the data supply ROL file is generated.

  All the data definition work for the report design is handled for you after you create your report and report field object definitions and generate the ROL file in Siebel Tools. In simple reports, the generated datastream is incorporated into your report design without modification, and you do not need to do anything else to set up data transfer between your Siebel application and Actuate.
- If you expand the page header component (ssFrmACTLISTHeader) and its child content frame (ssFrmBlueBack1), you can see how all the column heading labels and the blue line above them are defined. When you click a visual element in the layout pane or the structure pane, it is selected in both places. View the property list for one of the labels to see how its text, font, and other physical attributes are set up. Gray property text is the default from the parent object (ssLblHead in the sscustom.rol library); black text has been changed from the default.
- Click the Library Browser button. In the Choose Included Module window, double-click sscustom.rol. The listed components are library components; they serve as building blocks for your report. Browse through the list of library components in the sscustom library and compare them to similarly derived components in the report design.

  Generally, any items that you need to incorporate into a simple report are derived from generic components in the sscustom.rol library (except the datastream, which comes from the data supply library). These library components generally require only minor modification after they are introduced into the report design. In Actlist.rod, the labels, text controls, content frames, horizontal lines, and pagelist are derived from library components in sscustom.rol.

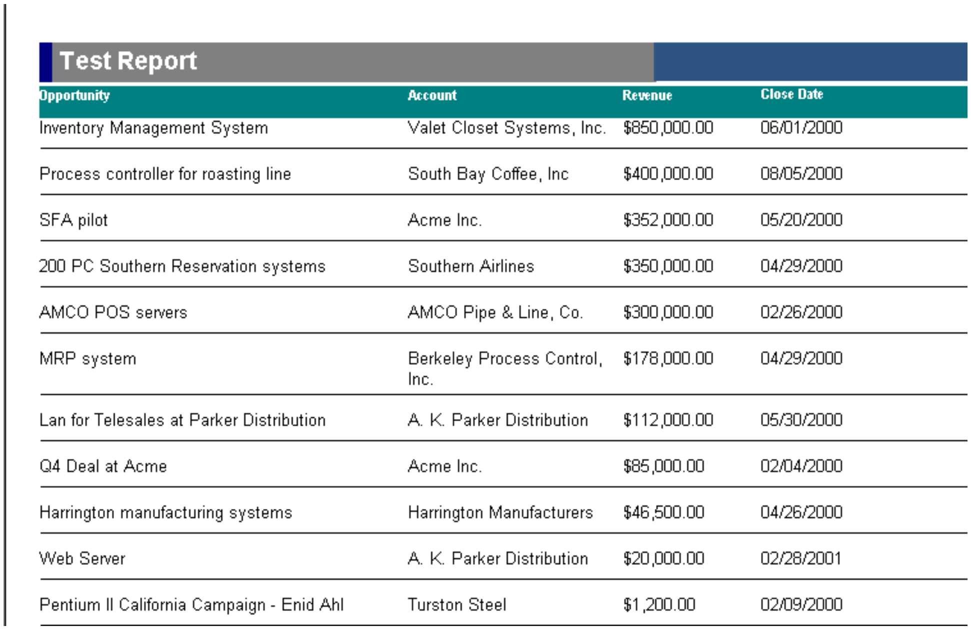
  You can now close or minimize the sscustom.rol Library Browser window.

- Expand the main content frame (ssFrmACTLISTContent). The child objects of this frame are all data controls. A data control displays the value in a field (or fields) obtained through the datastream. Open the Properties window for one of the data controls and note the setting in the ValueExp property. The ValueExp property holds the expression that determines what is displayed in the text control.

In the next section, you will create a similar report design yourself.

Example—Creating a Simple List Report

In this example, you create an opportunity list report. It lists the account name, opportunity name, expected revenue amount, and close date for every visible opportunity record. The resulting report will look like the one shown in Figure 8.



Test Report

Opportunity	Account	Revenue	Close Date
Inventory Management System	Valet Closet Systems, Inc.	$850,000.00	06/01/2000
Process controller for roasting line	South Bay Coffee, Inc	$400,000.00	08/05/2000
SFA pilot	Acme Inc.	$352,000.00	05/20/2000
200 PC Southern Reservation systems	Southern Airlines	$350,000.00	04/29/2000
AMCO POS servers	AMCO Pipe & Line, Co.	$300,000.00	02/26/2000
MRP system	Berkeley Process Control, Inc.	$178,000.00	04/29/2000
Lan for Telesales at Parker Distribution	A. K. Parker Distribution	$112,000.00	05/30/2000
Q4 Deal at Acme	Acme Inc.	$85,000.00	02/04/2000
Harrington manufacturing systems	Harrington Manufacturers	$46,500.00	04/26/2000
Web Server	A. K. Parker Distribution	$20,000.00	02/28/2001
Pentium II California Campaign - Enid Ahl	Turston Steel	$1,200.00	02/09/2000

Figure 8. Example Report

The following tasks are required to create the report:

- Create and export a new report object definition (and children) in Siebel Tools.
- Create a report design in Actuate.
- Add data control and label elements to the design.
- Compile and test the report.

Creating a New Report Object Definition in Siebel Tools

Each report design, whether custom or standard, normally has its own report object definition in Siebel Tools and a corresponding data supply library file. Report (and child) object definitions and exported datastream libraries are explained in "About Data Definition for Reports" on page 22.

In this example, you create a report object definition and children from scratch. An alternative approach is discussed in "Copying a Report Object Definition" on page 68.

To create a new report object definition (and children) in Siebel Tools

1. Open Siebel Tools and navigate to the Report object type in the Object Explorer.
2. Lock the Report project before proceeding to the next step. See Project in Step 4 for more information.
3. Click any record in the Object List Editor. Add a report record by choosing Edit > New Record.
4. In the new report record in the Object List Editor, enter properties as follows:
   - **Name.** Test Report

     This is the name used to refer to the report structure in Siebel Tools, for instance when you add the report to the Reports menu for a view.
   - **Project.** Report

     The Report project is standard for Siebel Sales. Report (SSV) is typically used for Siebel Service. Report projects are usually tied to specific Siebel applications.
   - **Access Base DB Name.** TESTREPT

     This specifies that Testrept.rox will be the name of the report executable invoked when this report object definition is invoked from the Reports menu in a view.

     NOTE: Capitalization is not required, although this value for standard reports is capitalized.
   - **Business Component.** Opportunity

     The business component specified here is one that provides the records for the main report. Subreports (in more advanced reports) have their own business components.
   - **Class.** CSSActuateReportViewer

     This specifies that the report is an Actuate report.
   - **Menu Text.** Test Report

     This is the text that will appear in the Reports menu for a view when this report is included in the view through the use of a view report object definition.

- **Template Name.** TESTREPT

  Specifies the name to be used for the generated data supply library file when the Generate Actuate Report option is invoked for this report object definition.

  NOTE: Capitalization is not required, although the values of this property for standard reports are capitalized.

5 Expand the Report object type in the Object Explorer tree and click the Report Field child object type to select it.

Add a report field object definition by choosing Edit > New Record. Click the Field property. In the Field pick applet, choose Name.

The Business Component property setting in the parent report object definition determines which fields can be included as report fields—namely, fields in that business component. Each report field object definition that you add as a child of the report defines a field that will be exported in the data supply library.

NOTE: A business component must be active before you can query a field in that business component. If the business component is not active, the field will not appear in the Reports Field dialog box.

Repeat this step for the Account, Revenue, and Close Date fields.

You can type in the name rather than looking it up in the pick applet. Match the spelling exactly.

6 Click the parent object definition (Test Report) in the upper Object List Editor window. Choose Tools > Utilities > Generate Actuate Report.

This generates a data supply ROL file, named according to the value in the Template Name property, in the \Siebdev\RPTSRC\LIB folder (or the equivalent on your system).

7 Navigate to the View object type in the Object Explorer.

8 In the Object List Editor, locate the Opportunity List View object definition. In the Object Explorer, expand View and click on View Report.

This is the My Opportunities view in the Opportunities screen. You will add the new report to the Reports menu for this view.

9 Add a view report object definition in the Object List Editor. Enter the name “Test Report” (match spelling and spacing exactly) and a sequence number that is greater by 1 than any other sequence number in the list.

The name that you specify identifies the report object definition that defines the report, including the menu text to display and the report executable to invoke. The sequence number specifies the report’s position in the menu relative to other reports.

10 Choose Repository > Compile and recompile the repository. Move the resulting SRF file to the \objects directory of your Siebel Dedicated or Mobile Web Client; for example \Siebdev\objects.

This last step is necessary for your Test Report to appear as a selection on the Reports button on the Opportunity List view. Repository compilation is not necessary for generating the data supply ROL file.

Creating a Report Design in Actuate e.Report Designer Professional

A report design file defines the layout and behavior for one report. In this exercise, a new report design is created from scratch, using library components.

To create a report design file in Actuate e.Report Designer Professional

1. Open Actuate e.Report Designer Professional.
2. Choose File > New. In the Create New Report window, select Blank Report and click OK.

   A blank design report appears.
3. Delete the components Report and SimplePageList in the Content and PageList sections of the blank report.

   For removal of Report and SimplePageList components, it is recommended that the components be deleted using the Project Browser:

   a. Navigate to View > Project Browser > Symbols.

   b. Delete the following components under the class NewReportApp Subclass Of AcReport:

      - Class Report Subclass of AcReportSection.
      - Class SimplePageList Subclass of AcSimplePageList.

      Deleting from the Project Browser makes sure that these components are completely deleted from the report.

   c. Close the Project Browser.
4. Choose Tools > Library Organizer.

   The Include Library window appears.

   a. Before adding the libraries remove the two existing sample libraries (erd_sample_library.rol and sample_sfdata_library.rol) and click More.

   b. Click Browse and navigate to \Siebdev\\RPTSRC\\LIB (or equivalent), choose sssiebel.rol, and click Open.

   c. Repeat this step for the files sscustom.rol, Testrept.rol, and sssiebel.bas.

   d. Click OK to close the Include Library window when all the files are added.

   e. Click OK to close the Library Organizer.

   The files sssiebel.rol and sscustom.rol are required libraries for all Siebel standard and custom reports. Testrept.rol is the data supply ROL file that you generated from Siebel Tools; it will define the way data is transferred to the new report.

   Components from afc.rol (and or components from the Actuate e.Report Designer Professional toolbar) and sssiebel.rol should not be used by report developers for designing Siebel-Actuate reports. When developing Siebel application reports, only use the sscustom.rol library.

5 Right-click on the top-level object (default name NewReportApp) and select Rename.

The Rename window appears.

- For the new name, input Testrept.

  It is recommended that the you use the same name you used for the ROD file saved previously (without the ROD extension).

6 Double-click the Testrept (top level) icon in the structure tree.

  a In the Component Editor window, click Class.

  b Change the Super Class value from AcReport to ssReport.

This is an important step. Siebel reports must inherit from ssReport (the Siebel standard report) instead of directly from AcReportSection (Actuate report). ssReport is provided in sscustom.rol and inherits from baseReport from sssiebel.rol.

NOTE: You cannot perform this step until you have included the sssiebel.rol library.

Two components are automatically added to the report when you change the report’s class: a report section (ssRpt) and a pagelist section (ssPageList).

7 While still in the Component Editor for the Testrept (top level) icon, click Properties.

- Scroll to the ssReportTitle property and enter the value Test Opportunity Report.

This value is used in the window title for the report table of contents in the report viewer.

8 Save the report design by choosing File > Save. Name the report Testrept.rod and specify a location where you want the report to be saved; for example:

\Siebdev\RPTSRC\custom\Testrept.rod

Remember to save the design file periodically as you work.

9 Make sure that the Show Empty Slots option in View > Options is checked.

You may want to deselect this option at a later time if you find the display of empty slots in the structure tree and layout pane distracting. For now, the empty slots must be visible because components will be dragged onto them from the sscustom library.

10 Expand the top-level report component.

  a Right-click the report section child component in the structure tree (ssRpt) and select Slot Information. The Single Structure Reference window appears.

  b Verify name is set to ssRpt.

  c Click Subclass and click close. The report section child component will now be named ssRpt1. You must subclass each component you introduce into the report if you plan on modifying the component. Otherwise, the system will not allow your changes.

11 Right-click the report section component (ssRpt1) and click Rename. Change the name to ssReportOpportunity.

It is good design practice to employ naming conventions for components in your reports and to give them meaningful names. The ssReport prefix indicates that this is a report section based on the ssReport library component. The Opportunity suffix identifies this as the master report section in which opportunity records are displayed.

12 Click the Library Browser button and choose Testrept.rol in the dialog box.

The Library Browser window opens, displaying the contents of Testrept.rol, which is the data supply library you generated in Siebel Tools.

13 Click the Library Browser button and open the sscustom.rol library. From the Library Browser window, drag and drop the ssFrm component onto the Content slot.

NOTE: When developing Siebel application reports, only use the sscustom.rol library. The use of Actuate components is not supported when modifying or developing reports.

This creates a frame that defines what appears in each report row. The frame is a child of the report section component.

a Right-click the new frame (ssFrm) and select Slot Information. On the Single Structure Reference window, click Subclass. Click Close.

b Right-click the same frame and select Rename. Rename it ssFrmOpportunityContent. Close the Library Browser window.

14 Drag and drop the datastream component (ssOpportunityQuery) from the Library Browser window onto the DataStream slot in the Design Editor. Close the Library Browser window.

The datastream component is now a child of the report section component. The datastream defines the way data is supplied to generate report rows. Each data record generates one report row in the parent report section.

When making changes in Siebel Tools for a report originally designed in Actuate e.Report Designer Professional and regenerating an ROL file, be aware of the following:

- Make sure that the ROD file that uses the ROL file is not open in Actuate e.Report Designer Professional.
- If any report fields are deleted, make sure to remove the fields from your Actuate report as well.
- Changes made to the ROL file are valid only for that instance of the ROL file. The changes will need to be entered again. However, if the changes are locally subclassed, they are safely stored in the ROD file.

Adding Label and Data Elements to the Design

At this point, you have created the report design file and three of its key structural components: the report design (root), the report section, and the pagelist. You have also obtained the datasource, by reference, from the datasource library file that you previously generated, and put it in the report section whose data it will supply. Now you create a frame for the page header to define the elements that appear at the top of each page, and a main content frame to define the elements that appear in each report row.

To add text and label elements to the design

1. Click the Library Browser button and open the sscustom.rol library. From the Library Browser window, drag and drop the ssFrm component onto the Content slot.

   NOTE: When developing Siebel application reports, only use the sscustom.rol library. The use of Actuate components is not supported when modifying or developing reports.

   This creates a frame that defines what appears in each report row. The frame is a child of the report section component.

   a. Right-click the new frame (ssFrm) and select Slot Information. On the Single Structure Reference window, click Subclass. Click Close.

   b. Right-click the same frame and select Rename. Rename it ssFrmOpportunityContent.

2. Drag and drop the ssTxt component from the Library Browser window onto the empty Content child slot of the content frame (ssFrmOpportunityContent).

   The Component Properties dialog box appears.

3. In the Component Properties dialog box, select the ssName field from the datastream by clicking it in the drop-down list, then click OK to dismiss the Component Properties dialog box.

   You specify the source of data for a data control in the ValueExp property.

4. Resize the new data control on the layout grid to make it the appropriate size for name data (if it is too narrow, the name will be truncated).

   Right-click the data control and select Rename in the menu. Give the text control a unique descriptive name, in this case txtOpportunity.

   You generally obtain the prefix in a report design component from somewhere in the name of its parent library component. The txt prefix (from ssTxt) is a reasonable standard for dynamic data controls derived from ssTxt and is consistent with naming in standard reports.

5. Right-click the data control and select Properties in the menu. Check the CanGrow property to verify that it is set to TRUE.

   When this property is TRUE, multiple-line values in the business component fields print as multiple-line values in the report. When it is FALSE, only the first line of each value prints.

   NOTE: When possible, use the Default button in the Component Editor window to reset a property such as CanGrow to its default (TRUE in this case) rather than entering the value manually. This helps to keep the inheritance intact by eliminating unnecessary subclassing.

6. Repeat Step 4 on page 65 through Step 8 to add the following:

   - A control called txtAccount based on the ssAccount datastream variable
   - A control called txtRevenue based on ssRevenue_Formatted
   - A control called txtCloseDate based on ssClose_Date

   Note that you drop the ssTxt component on the parent content frame (ssFrmOpportunityContent) because there are no empty child slots for the purpose. Also note that you need to reposition each data control in the layout pane to the right after it is created.

7 Drag and drop the ssFrmBlueBack library component onto the empty page header slot. Subclass the new frame component (using the subclassing instructions from "Creating a Report Design in Actuate e.Report Designer Professional" on page 62) and rename it ssFrmOpportunityHeader.

The page header will contain the column headings. Bold white column heading labels on a blue-green background are standard in Siebel reports and give your custom report the same appearance.

8 Drag and drop the ssLblHead component onto the page header frame (ssFrmOpportunityHeader), once each to create the labels indicated below. Set the Text property for each label as indicated. You will need to resize, reposition, and rename each label after it is created. Each label should be aligned vertically with the corresponding data control:

- A label called lblOpportunity, with a Text property value of Opportunity
- A label called lblAccount, with a Text property value of Account
- A label called lblRevenue, with a Text property value of Revenue
- A label called lblCloseDate, with a Text property value of Close Date

9 Set the AlternateLines attribute of the ssFrmOpportunityContent frame to 1. A line separator is automatically added to each displayed content row frame.

The separator line creates a demarcation between report rows.

10 Right-click the top-level report component and choose Properties. Enter a value of Test Report in the ssReportTitle property.

This defines the title to be printed in the blue-green area on the upper left.

Compiling and Testing the Report

The report must be built and compiled in Actuate e.Report Designer Professional for you to run it. Then it can be debugged locally in Actuate e.Report Designer Professional and, when it is ready to be deployed, the executable version of the report is moved to the folder where Siebel applications obtain their report executables.

To compile and install the report

1 Open Siebel Sales.

2 From the application-level menu, choose View > Site Map > Opportunities > My Opportunities.

Make sure that the desired query is active.

The appropriate application and view must be open for the report to obtain records. Also, the set of records displayed in the view is the set of records that is used in the report.

3. In Actuate e.Report Designer Professional, click the Run button.

   The report appears in the Web browser. You can test your report as you develop it in Actuate. Note that the report displays the data records in the same order in which they are displayed in the Siebel application. If the Actuate Output window displays an error message indicating that no data is available for the report, make sure that your Siebel application has queried the appropriate records and that the cursor is positioned on the first record.

   NOTE: If the cursor is positioned on a record other than the first one, not all records will be displayed. This is because of the Forward Only mode of the query.

4. Make corrections to the report design as necessary and recompile as often as required.

   NOTE: While making corrections to the report, if you need to review the Actuate Output window for the previous error message, go to View > Output Window or use the hotkey Alt-v followed by w.

5. Use Microsoft Windows Explorer to copy the Testrept.rox file from the location where you saved the testrept.rod file; for example, C:\Siebdev\RPTSRC\custom to the REPORTS folder in your Siebel application installation directory; for example, C:\Siebel\REPORTS.

6. In Siebel Sales, click the Reports button and select Test Report from the drop-down list.

   The report should appear in Siebel Sales. If there are problems, verify the following:

   - If there is no Test Report option in the Reports menu, verify that the My Opportunities view is active. If the correct view is active, your problem lies in the configuration of the view report object definition that defines this menu option. Make sure that it is a child of the correct view (Opportunity List view).

   - If a message appears that the Siebel application cannot find the report, verify that you have moved the executable to the correct folder.

Compiling Reports with Actuate 7

The Actuate 7 e.Report Designer Professional (erdpro) compiler includes changes made to improve performance. The compiler is now more stringent and catches errors that might have been ignored in earlier versions. When creating new reports with Actuate 7 erdpro, the report compilation might fail due to the *Indirect DataRow Member Access* variable not being set correctly. Follow these instructions to set or correct this variable.

To compile reports with Actuate 7

1. Edit the following registry variable on the system where erdpro is installed:

   ```
   HKEY_CURRENT_USER\Software\Actuate\e.Report Designer Professional
   7.0\Settings\IndirectDataRowMemberAccess
   ```

2. Set value to 1 (default value is zero).

3. If `IndirectDataRowMemberAccess` is not present in your registry, create a `DWORD` value with the name `IndirectDataRowMemberAccess` and Hexadecimal value of 1.

4. Close all open erdpro instances and open a new erdpro session.

5 Compile your report.

Alternative Report Creation Strategies

The procedure outlined in the preceding sections is one approach to creating a new custom report. There are alternative approaches to part or all of the process. Some of these are explained briefly in the sections that follow.

Copying a Report Object Definition

The procedure for creating a new report object definition (and children) in Siebel Tools, described in "Creating a New Report Object Definition in Siebel Tools" on page 60, may prove cumbersome when a new set of object definitions needs to be created that is very similar to an existing one, and the existing one is complex. An alternative is to copy the desired report object definition. This creates a new one with the same property settings, child report field, and subreport object definitions.

To copy an existing report object definition and children

1. In Siebel Tools, navigate to the Report object type and the report object definition you want to copy.
2. Lock the Report project (choose Repository > Lock Project).
3. Choose Edit > Copy Record.

   This creates a new report object definition with the same set of children and duplicated property settings. The exceptions are the Menu Text and Name properties, which you must specify. You should also change the Template Name and Access Base DB name settings, so that the data supply library and executable report have different names from those in the original.
4. Make alterations as necessary to the new report object definition and children.
5. Export to a data supply library file by choosing Tools > Generate Actuate Report.
6. Unlock the Report project.

Copying a Report Design

An Actuate report design (ROD) file can be copied to a new file (with a different name) using Windows Explorer or the File > Save As option in Actuate e.Report Designer Professional. Actuate treats the new design as independent from the original, but maintains all relationships with included libraries.

This is a desirable approach when you wish to reuse many of the design elements in an existing report design, but have various modifications to make. This is a common situation, and you will find this technique valuable.

After you copy an existing report design, make the following modifications to the new report design:

- Make a copy of the original report design definition with a new name, as described in "Copying a Report Object Definition" on page 68. Remove the existing datastream from the report design, and drag and drop the new one onto the report design from the new data supply library file. This is an important step because it is impractical to have two report designs sharing a single datastream and a single report object definition.

  NOTE: When copying a report design, make sure that the datastream being added is using the same business component as the report object definitions for the new report. Errors will occur if the datastream being replaced is based on a different business component than the datastream that is being added to the new report.

- Make sure that the report object definition has the correct settings in the Template Design, and that the Access Base DB Name properties correspond to the new report design.
- Rename the top-level report component. This is not a critical step, but it helps you orient yourself when you are working on each report in Actuate e.Report Designer Professional.

Using a Custom Component Library

If you create a design component and child components that will prove useful in similar reports, you can create a custom component library that allows you to reuse these components. An example of a custom component library is the ssQuote.rol library used in various quote reports, including Quotestd.rod, Quotepro.rod, and Quosum.rod.

Typically, this technique is used to reuse page header, page footer, or content frames containing a number of labels and data controls, but any component and its children can be published in this fashion for reuse.

To create a custom component library

1. Open the report design that contains the components you want to publish in the custom library.
2. Choose Tools > Library Organizer.

   The Library Organizer window appears.

   a. Click New.
   b. Specify the destination folder (this should be C:\Siebdev\RPTSRC\ENU\LIB or equivalent) and filename (the filename should begin with ss, to designate it as a Siebel custom library and distinguish its name from those of the datastream libraries). Click Save.

      If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

      See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

   c. Click OK on the Library Organizer window.

   The Library Browser window opens for the new library, containing only the top-level library component.

3. Drag and drop the desired parent component from the report design (such as a content frame) onto the top-level library component.

   The Component Drop dialog box appears.

4. Click the Publish the component radio button and click OK.

   The component that you published and its children are added to the library. Their names in the report design file are changed to light gray, indicating that they are now subclassed components from the version in the library.

Additional Information for Developing and Deploying Siebel Reports

This section includes various additional issues that pertain to developing and deploying reports in Siebel applications.

Migrating Custom Reports

Migrating custom reports involves a short process to complete. The following steps will need to be completed for each custom report you have created.

Before starting, make sure that the latest release of Actuate e.Report Designer Professional is installed on your machine. The following upgrade instructions are specifically for Siebel Reports 6.0 and later.

To upgrade a custom report to Siebel release 7.x for Actuate 7

1. Using an existing custom report from Siebel release 6.0 and later, retrieve an ROD (report design) file.
2. Generate an ROL (report library) file from Siebel Tools.
   a. In Siebel Tools, navigate to Reports and select a desired report.
   b. From the Tools menu, select the Generate Actuate Report option.
3. Place the ROD file and the generated ROL file (and other necessary libraries) in a directory on your machine.
4. Open the ROD file in Actuate e.Report Designer Professional.
5. Compile the report to generate an ROX (report executable) file by selecting the Build option from the Report menu.

NOTE: Regarding upgrading of the sscustom.rol file from one release to the next, the format of the file is upgraded, but any custom changes are not. After an upgrade you have to manually add any changes and adjustments to the new version of the sscustom.rol.

Moving ROX files to the Reports Server

The ROX files for all custom reports need to be manually moved to the Reports Server.

To move the ROX files to the Reports Server

1. Log in to Actuate Management Console as Administrator.

   NOTE: A password is usually not necessary. If one is required, check with your Siebel Application developer for the password selected during the Reports Server installation.

2. Select Files & Folders.
3. Select the folder that you want to add an ROX file (for example, the Siebel Reports folder) by placing a check mark in the box to the left of the name of the folder.

   NOTE: The word Folder will appear in the Type column.

4. Click Add File.

   The ROX file that you want to add should reside on your local machine from where you are launching Management Console.

5. Browse to select the ROX file and provide the information requested.
6. If the file already exists:
   1. Select one of the following:
      - Replace the latest version.
      - Create a new version.
      - Keep only a specified number of versions.

      Type the number of versions that you want to keep of this ROX file.
   2. Select the properties (Description, Permissions, or Auto archive rules) from the latest version of the file that you want applied to this ROX file.

About Using a Datastream Twice

Two sequential report sections can use the same datastream. However, this approach generates the following error message, Operation is not allowed on sql object in forward only mode. The workaround is to comment out the SetForwardOnly mode statement in the ROL’s Fetch method. This approach will negatively impact the performance of the report. The datastream is read once each time it is executed.

Another approach is to retrieve the data once and store the rows in a global list variable that can be used as the datasource of subsequent report sections. The steps to accomplish this, with a few modifications, are described in Chapter 10, “Sorting Records in Memory.”

About sssiebel.bas and Migration Considerations

The sssiebel.bas BASIC file must be included in all report design files for Siebel reports. This is because some object interface methods are in use that are referenced in this file.

NOTE: In general, the sssiebel.bas file is automatically loaded when the sssiebel.rol file is loaded for a report. A manual loading of this file is usually not necessary. However, in some cases (such as in legacy design or a lost association with sssiebel.rol), the user may need to include sssiebel.bas if indicated by Actuate.

Another important migration consideration is that data supply ROL files must be generated from Siebel Tools version 7.x to work correctly with server reporting. You must regenerate the data supply ROL files for all your custom report design files.

About Backing Up Report Design and Library Files

You should create a set of development directories for report source files—particularly report design files and library files—to make sure that older versions are not overwritten. Ideally, you should employ source code control software for this purpose. When modifications are made repeatedly to a single set of source files in one location, possibly by multiple developers, you risk the loss of report design and development effort. A system for backing up source files and retaining earlier versions significantly reduces this risk.

About Emailing a Report

You can send a report instance (ROI) file to another Siebel user for viewing. The recipient opens the Srviewer.exe application (located in C:\Siebel\bin), clicks the Open File button, and navigates to the desired ROI file. This opens the report instance in the Actuate Report Viewer.

It is not necessary for a Siebel application to be running or for the recipient to have the view or data used in generating the report. This is because an ROI file is self-contained.

However, the Srviewer.exe program cannot be sent to a non-Siebel user. It requires that certain DLL files be installed as well as the executable.

For non-Siebel users who require electronic copies of report output, you can employ one of these alternatives:

- Create a report that looks acceptable in HTML and send it as an HTML file. Be aware that there are limitations such as printability, the entire report being a single HTML page, and so on.
- If the report output is present on the Reports Server, download it in PDF format and send the file.
- For local reports, if you have the appropriate Adobe PDF printer driver installed, you can generate a PDF file by printing with that driver and then emailing the resulting file.

NOTE: To enable email notification from the Actuate iServer System, Microsoft Exchange Server must be used as the email Server. Email notification will not work if a standard SMTP email server is used. For more information on enabling email notification, see the *Administering Actuate iServer System* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

About Printing

If you set the paper size in the report design, the setting is ignored when the report is printed. The report will always use the default setting in the print setup. Also, the setting in the report design can only take effect if the Actuate API is used. However, the use of Actuate APIs is not recommended in Siebel eBusiness applications.

Also, if the default page size is changed on the Page Setup from Letter to Legal, when in the report viewer, the change will only last as long as your PC is not rebooted or restarted. If you restart, the default paper size reverts back to Letter page size. This behavior is relevant in the Mobile Web Client environment, which uses the Siebel Report Viewer.

For more information regarding printing, see the *Using e.Reports* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

Using Actuate e.Report Designer to Create Reports

Actuate e.Report Designer allows you to design and build reports using its graphical user interface. This application complements Actuate e.Report Designer Professional and is used by business users to modify and distribute a variety of reports. No programming is required, therefore Actuate e.Report Designer is not used to create Siebel Actuate reports. This application supports both modifying complex reports and using components from libraries.

Creating Reports Using Siebel Report Libraries

To create reports using Siebel Report Libraries (sssiebel.rol file) in Actuate e.Report Designer, perform the following tasks.

The Actuate solution to enable Siebel reporting in Actuate e.Report Designer is based on two files, siebel_report_building.xml and wizard.rod.

These files reside on the system drive. The following locations are examples of where to locate these files:

- C:\Program Files\siebel\7.x.x\Tools\RPTSRC\siebel_report_building.xml
- C:\Program Files\siebel\sea7.x.x\Tools\RPTSRC\STANDARD\wizard.rod

Using these files, administrators can use Actuate e.Report Designer without any additional changes.

You can also develop simple reports in Actuate e.Report Designer, based on new or existing reports.

To create a new report in Actuate e.Report Designer

1. Open Actuate e.Report Designer.
2. Navigate to the siebel_report_building.xml file:

   a. From the menu, choose View > Options.

  b. Click General and in the Configuration file section, click browse (...).

  c. From the Locate Configuration File window, navigate to the siebel_report_building.xml file and click Open.

3. While still in the Options window, choose the Global Search Path.

   a. Click New.

   b. Click browse (...) and navigate to the following location:

      C:\Siebdev\RPTSRC\language\LIB

      where:

      language = the Siebel code for the Language Pack you installed; for example, enu for U.S. English.

   c. Click Open.

   d. Click OK to close the Options window.

4. Select File > New > Siebel Report Template.

5. Include custom libraries and controls in the same manner as for Actuate e.Report Designer Professional.

To enable an existing report in Actuate e.Report Designer

1. To enable existing projects, first perform Step 1 through Step 3 from "Creating Reports Using Siebel Report Libraries" on page 73.

2. From the menu, choose File > Open to select an existing Siebel report.

   NOTE: There is no way to recompile ROX files separately with the exception of clicking the Run button from the tool bar in Actuate e.Report Designer.

   Custom controls and custom code must be added in Actuate e.Report Designer Professional first before using them in Actuate e.Report Designer.

Adding Sorting to Reports

You can add sorting and grouping capability to a report designed with Actuate e.Report Designer, by using a custom ROL prepared in Actuate e.Report Designer Professional (for example, aclist_sorteable.rol for account list report). The ROL file is based on slightly modified Siebel Tools-generated ROL, which includes ssMemoryDataSorter on top of an existing DataStream (see Figure 9). This allows you to build entire reports enabled with wizard-based sorting and grouping.

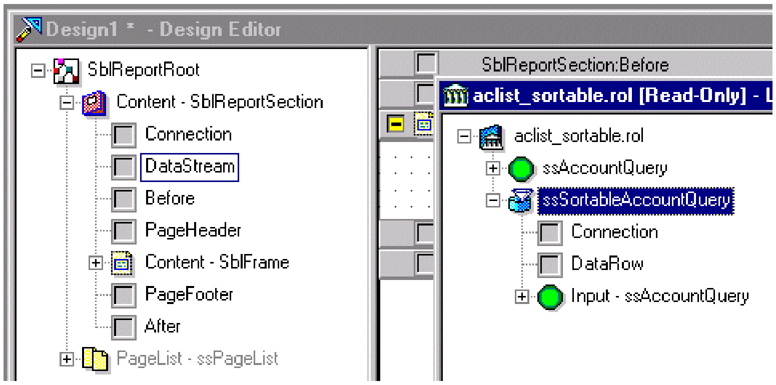


Figure 9. Account List Report ROL File

To add sorting to a report

1. Remove the dummy (or existing DataStream) data source.
2. Choose Tools > Library Organizer and include aclist_sorteable.rol (as an example) that was created in Actuate e.Report Designer Professional.
3. Drag and drop the Memory Sort control in the data stream slot of the Report section.
4. Choose Tools > Sorting & Grouping.
5. The Grouping tab of the Sorting and Grouping dialog box displays the fields available for grouping.

   These are fields in those report tables and views that have not yet been specified as group keys.
6. On the left side of the dialog box, double-click the field to group, or select the field and click the greater than symbol (>).
7. Actuate e.Report Designer adds a grouping field, and removes the field from both the Available Fields list and the detail frame of the report.

7 Reports with Group Sections

This chapter discusses the use of group sections to create reports with group breaks. Group breaks are the points in a list of records where the value changes in a key field, resulting in special processing—usually including the printing of a new heading, and sometimes a page break. For example, in a list of accounts, you may want a heading displaying the account name before the group of records for each account. A change in the account name field between records triggers a new heading.

This chapter consists of the following topics:

Using Group Sections Overview

In Siebel reports, group break behavior is implemented through the use of a group section component in the report design.

Note that the business component records must be sorted with the group field as the primary sort key (generally in the Sort Specification property of the report object definition, if it is different from the default sort order for the business component). Otherwise, the records do not group properly.

Note also that a group section is for clustering records of a single business component, based on a field or fields in that business component. This is a different scenario from listing detail records of another business component for each master business component record. The latter scenario requires configuration of a master-detail report, as described in Chapter 8, "Master-Detail Reports." This does not preclude the use of group sections within a master-detail report, either at the master level or in one or more of the subreports.

This following examples represent reports using single group sections. Group sections can also be placed inside another group section, known as nesting groups. Nesting groups are created by placing the outermost group section in the Content slot of a report section. For more information, see the section on working with sections in the *Developing Advanced e.Reports* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

How a Group Section Works

Figure 10 illustrates the structure of a simple report that employs a group section.

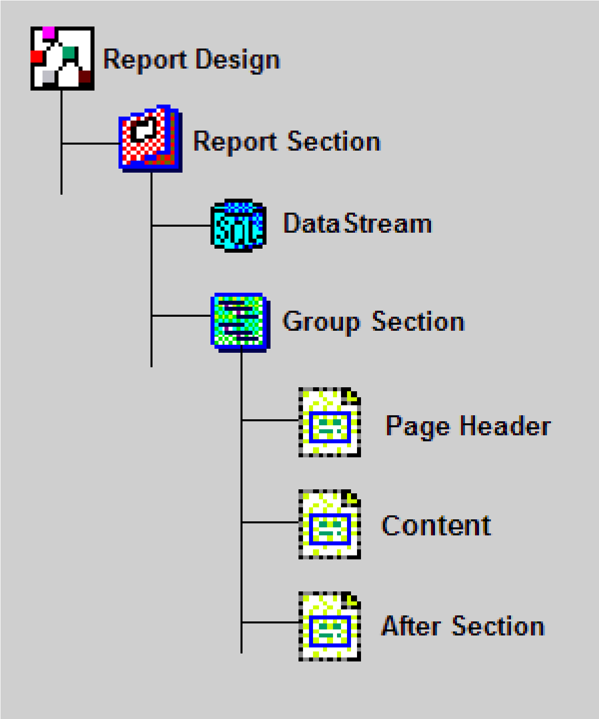


Figure 10. Group Report Structure

This includes the following major components:

- **Report Design.** This is the top-level component in a report; it corresponds to the ROD file in which it resides. There are no special features of the report design component for group reports.
- **Report Section.** A report section groups together components that define the source of data, physical layout, and behavior of a master report or subreport. The group section is a child of the report section whose records the group section will cluster.
- **DataStream.** The datastream component defines the source of data for the report section. There are no special features of a datastream for a grouped report section, other than the requirement that the data be sorted with the group field as the primary sort key.
- **Group Section.** The group section component implements grouping and group break behavior for the records in its parent report section. The group section has a Key property that defines the field that determines how the records are grouped, and other properties that define grouping behavior, such as whether each group break causes a page break.

- **Page Header Frame.** In a report section with a group section, the page header frame is a child of the group section, not of the report section. The contents of the page header frame are displayed at the beginning of each new group, in addition to the beginning of each page. The page header contains group break information, such as the sales stage name for each group of opportunities by sales stage.
- **Content Frame.** The content frame defines the layout of one report row, as it does in a nongrouped report section. However, it is a child of the group section, rather than of the report section.
- **After Section.** The After section defines what appears in the report following each group. This section can be omitted or used to display group totals. See "Adding Group Totals to Reports" on page 89 for details.

Examining a Report with a Group Section

To learn more about the configuration of a simple grouped report, it is helpful to open a standard Siebel report of this type in Actuate e.Report Designer Professional and study it. A good report to study for this purpose is the Contacts By State parameterized report, which is invoked from a view in the Contacts screen (typically the My Contacts view). You should examine the report output in Siebel Sales, and the report design in Actuate e.Report Designer Professional, leaving both open to compare them.

To generate the Contacts By State report in Siebel Sales

1. Open Siebel Sales.
2. Click the Contacts screen tab.
3. Select the Contacts List on the link bar.

   My Contacts is the default view.
4. While in the My Contacts view, click the Reports button and from the drop-down menu select Contacts By Category.

   The Contact Parameters window appears.
5. From the Sort By drop-down menu, select State, and click Finish.

   The Contacts By State report appears.

To open the report design for the Contacts By State report

1. Open Actuate e.Report Designer Professional.
2. Choose File > Open.

3 In the Open dialog box, navigate to the \Siebdev\\RPTSRC\\ENU\\STANDARD folder (or the equivalent on your computer) and choose cntstate.rod.

If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \\ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \\DEU for Germany.

See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

Figure 11 shows the cntstate.rod report design file in Actuate e.Report Designer Professional.

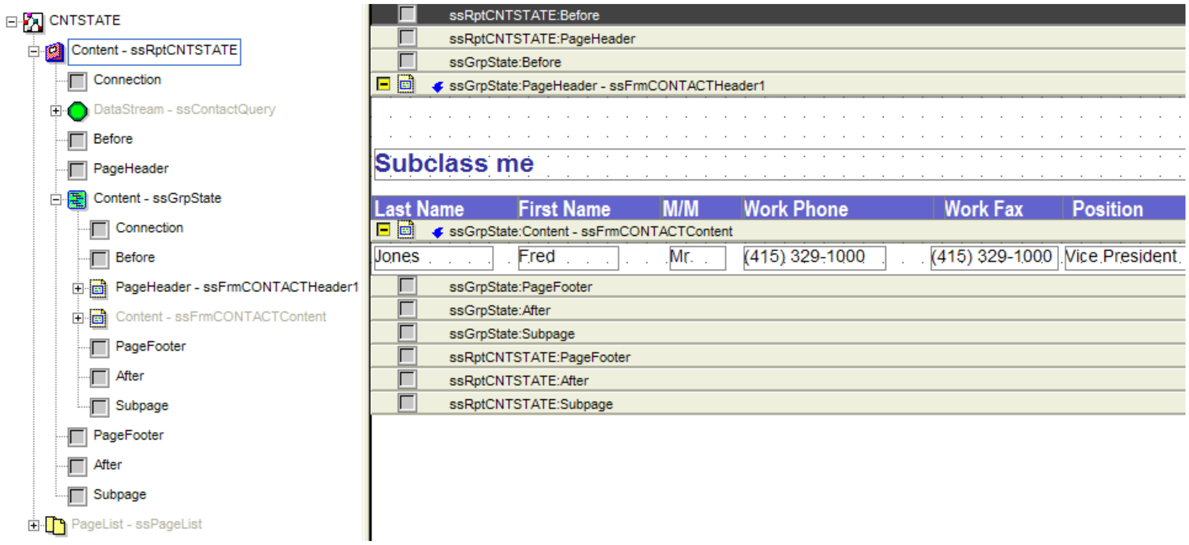


Figure 11. Contacts By State Report Design

Explore the report design, compare design elements to the corresponding features in the report output, and right-click components to view their property lists.

Notice the following features of this report design:

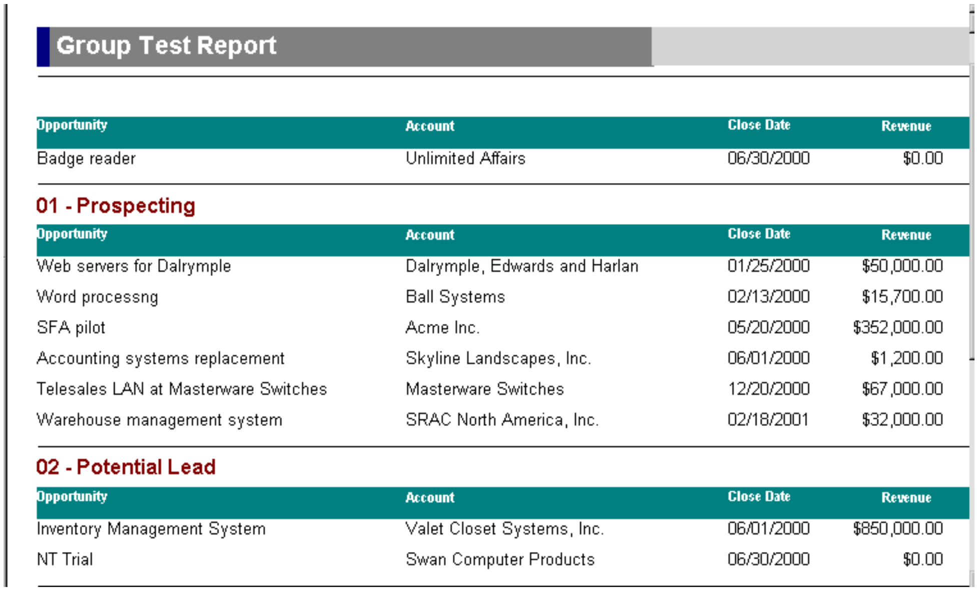
- The group section component, ssGrpState, has child page header and content frames. These are ssFrmCONTACTHeader1 and ssFrmCONTACTContent. These frames are subclassed from the sscntct.rol custom component library because the same header and content frames are used in most of the contact reports. For more information on custom component libraries, see "Using a Custom Component Library" on page 69.
- If you expand the page header component (ssFrmCONTACTHeader1), you will see a child content frame (ssFrmBlueBack1). If you expand the child content frame, you see the column heading labels that appear in the page header.
- Another child component of the page header is a data control called ssTxtCONTACTHeader1. When you select this control in the component tree, it is also selected in the layout pane (the sample text in the control says Subclass Me).

- The page header data control (ssTxtCONTACTHeader) displays the state name abbreviation for each new state. You can see this in the generated report in Siebel Sales. The property in the data control that configures this behavior is ValueExp. The value of this property is ssState, which is the datastream variable corresponding to the State field in the Contact business component.
- If you examine the properties in the group section component (ssGrpState), you will notice that it has a Key property and that this property has a setting of ssState. The Key property value in the group section component determines when a sort break occurs, namely when there is a change in the value in the corresponding field between business component records. The change in group key values triggers the redisplay of the page header with new values.

In the next section, you will create a similar report design yourself.

Example—Creating a Report with a Group Break

In this example, you create an opportunity list report in which the opportunity records are grouped by sales stage. The report lists the opportunity name, account name, expected revenue amount, and close date for each opportunity record. It also displays the sales stage in the page header before each group of opportunity records with that sales stage. As stated, the resulting report will look like the one in Figure 12.



Group Test Report

Opportunity	Account	Close Date	Revenue
Badge reader	Unlimited Affairs	06/30/2000	$0.00

01 - Prospecting

Opportunity	Account	Close Date	Revenue
Web servers for Dalrymple	Dalrymple, Edwards and Harlan	01/25/2000	$50,000.00
Word processng	Ball Systems	02/13/2000	$15,700.00
SFA pilot	Acme Inc.	05/20/2000	$352,000.00
Accounting systems replacement	Skyline Landscapes, Inc.	06/01/2000	$1,200.00
Telesales LAN at Masterware Switches	Masterware Switches	12/20/2000	$67,000.00
Warehouse management system	SRAC North America, Inc.	02/18/2001	$32,000.00

02 - Potential Lead

Opportunity	Account	Close Date	Revenue
Inventory Management System	Valet Closet Systems, Inc.	06/01/2000	$850,000.00
NT Trial	Swan Computer Products	06/30/2000	$0.00

Figure 12. Example Report with a Group Break

The following tasks are required to create the report:

- Create and export a new report object definition (and children) in Siebel Tools.
- Create a report design in Actuate e.Report Designer Professional.

- Add data control and label elements to the design.
- Compile and test the report.

Creating a New Report Object Definition in Siebel Tools

Each report design normally has its own report object definition in Siebel Tools and a corresponding data supply library file. In Chapter 6, "Creating a Simple List Report," the report object definition was created from scratch. In this example, you copy an existing report object definition (and children) and change some property values.

To create a new report object definition in Siebel Tools by copying

1. In Siebel Tools, navigate to the Report object type in the Object Explorer and select the Opportunity List - Current Query report object definition in the Object List Editor.

   Lock the Report project (choose Tools > Lock Project).

2. Choose Edit > Copy Record.

   This creates a new report object definition, with the same set of children as the original, and duplicated property settings.

3. Change the following property settings in the new report object definition (the other property settings do not need to be changed):

   - **Name.** Test Group Report

     This is the name used to refer to the report structure in Siebel Tools, for instance when you add the report to the Reports menu for a view.

   - **Access Base DB Name.** GROUPRPT

     This specifies that GROUPRPT.rox will be the name of the report executable invoked when this report object definition is invoked from the Reports menu in a view.

   - **Menu Text.** Sales Stage - Test

     This is the text that will appear in the Reports menu for a view when this report is included in the view through the use of a view report object definition.

   - **Template Name.** GROUPRPT

     Specifies the name to be used for the generated data supply library file when the Generate Actuate Report option is invoked for this report object definition.

   - **Sort Specification.** Sales Stage

     Sorts the opportunity records into sales stage order, which is necessary for the group breaks to work correctly. This specification is translated into corresponding code in the datastream methods.

4 Export to a data supply library file by choosing Tools > Utilities > Generate Actuate Report.

This generates a data supply ROL file, named according to the value in the Template Name property, in the *Siebdev*\RPTSRC\LIB folder (or equivalent on your system).

5 Unlock the Report project.

6 Navigate to the View object type in the Object Explorer.

7 In the Object List Editor, locate the Opportunity List View object definition. In the Object Explorer, expand View and click on View Report.

This is the My Opportunities view in the Opportunities screen.

8 Lock the Oppty (SSE) project.

9 Add a view report object definition in the Object List Editor. Enter the name "Test Group Report" (match spelling and spacing exactly) and a sequence number that is 1 greater than any other sequence number in the list.

10 Select Repository > Compile and recompile the repository. Move the resulting SRF file to *Siebdev*\OBJECTS\ENU (or the equivalent location on your system).

11 Unlock the Oppty (SSE) project.

Creating a Report Design in Actuate e.Report Designer Professional

In this exercise, a new report design is created from scratch, using library components.

To create a report design file in Actuate e.Report Designer Professional

1 Open Actuate e.Report Designer Professional.

2 Choose File > New. In the Create New Report window, select Blank Report and click OK.

A blank design report appears.

3 Delete the components Report and SimplePageList in the Content and PageList sections of the blank report.

For removal of Report and SimplePageList components, it is recommended that the components be deleted using the Project Browser:

a Navigate to View > Project Browser > Symbols.

b Delete the following components under the class NewReportApp Subclass Of AcReport:

❏ Class Report Subclass of AcReportSection.

❏ Class SimplePageList Subclass of AcSimplePageList.

Deleting from the Project Browser makes sure that these components are completely deleted from the report.

c Close the Project Browser.

4 Choose Tools > Library Organizer.

The Include Library window appears.

a Before adding the libraries remove the two existing sample libraries (erd_sample_library.rol and sample_sfdata_library.rol) and click More.

b Click Browse and navigate to \Siebdev\\RPTSRC\\LIB (or equivalent), choose sssiebel.rol, and click Open.

c Repeat this step for the files sscustom.rol, Grouprpt.rol, and sssiebel.bas.

d Click OK to close the Include Library window when all the files are added.

e Click OK to close the Library Organizer.

The files sssiebel.rol and sscustom.rol are required libraries for all Siebel standard and custom reports. Grouprpt.rol is the data supply ROL file that you generated from Siebel Tools that defines the way data is transferred to the new report.

Components from afc.rol (or components from the Actuate e.Report Designer Professional toolbar) and sssiebel.rol should not be used by report developers for designing Siebel-Actuate reports. When developing Siebel application reports, only use the sscustom.rol library.

5 Right-click on the top-level object (default name NewReportApp) and select Rename.

The Rename window appears.

- For the new name, input Grouprpt.

  It is recommended that you use the same name you used for the ROD file saved previously (without the ROD extension).

6 Double-click the Grouprpt (top level) icon in the structure tree.

a In the Component Editor window, click Class.

b Change the Super Class value from AcReport to ssReport.

This is an important step. Siebel reports must inherit from ssReport (the Siebel standard report) instead of directly from AcReportSection (Actuate report). ssReport is provided in sscustom.rol and inherits from baseReport from sssiebel.rol.

NOTE: You cannot perform this step until you have included the sssiebel.rol library.

Two components are automatically added to the report when you change the report's class: a report section (ssRpt) and a pagelist section (ssPageList).

7 While still in the Component Editor for the Grouprpt (top level) icon, click Properties.

- Scroll to the ssReportTitle property and enter the value Group Opportunity Report.

This value is used in the window title for the report table of contents in the report viewer.

8 Save the report design by choosing File > Save. Name the report Grouprpt.rod and specify a location where you want the report to be saved; for example:

\Siebdev\\RPTSRC\\custom\\Grouprpt.rod

Remember to save the design file periodically as you work.

9. Make sure that the Show Empty Slots option in View > Options is checked.

   You may want to deselect this option at a later time if you find the display of empty slots in the structure tree and layout pane distracting. For now, the empty slots must be visible because components will be dragged onto them from the sscustom library.

10. Expand the top-level report component.

    a. Right-click the report section child component in the structure tree (ssRpt) and select Slot Information.

       The Single Structure Reference window appears.

    b. Verify that the name is set to ssRpt.

    c. Click Subclass and click Close.

       The report section child component will now be named ssRpt1.

       You must subclass each component you introduce into the report if you plan on modifying the component. Otherwise, the system will not allow your changes.

11. Right-click the report section component (ssRpt1) and click Rename. Change the name to ssRptOpportunity.

    It is good design practice to employ naming conventions for components in your reports and to give them meaningful names. The ssReport prefix indicates that this is a report section based on the ssReport library component. The Opportunity suffix identifies this as the master report section in which opportunity records are displayed.

12. Expand the report section component (ssRptOpportunity). Click the Library Browser button and choose sscustom.rol. Drag and drop the ssGrp library component onto the Content child slot of the report section.

13. Click the Library Browser button and choose Grouprpt.rol in the dialog box.

    The Library Browser window opens, displaying the contents of Grouprpt.rol, which is the data supply library you generated in Siebel Tools.

14. Click the Library Browser button and open the sscustom.rol library. From the Library Browser window, drag and drop the ssFrm component onto the Content slot.

    NOTE: When developing Siebel application reports, only use the sscustom.rol library. The use of Actuate components is not supported when modifying or developing reports.

    This creates a frame that defines what appears in each report row. The frame is a child of the report section component.

    a. Right-click the new frame (ssFrm) and select Slot Information. On the Single Structure Reference window, click Subclass. Click Close.

    b. Right-click the same frame and select Rename. Rename it ssFrmOpportunityContent. Close the Library Browser window.

15 Drag and drop the datastream component (ssOpportunityQuery) from the Library Browser window onto the DataStream slot in the Design Editor. Close the Library Browser window.

   The datastream component is now a child of the report section component. The datastream defines the way data is supplied to generate report rows. Each data record generates one report row in the parent report section.

16 With the sscustom.rol library still open from the Library Browser window, drag and drop the ssGrp component onto the group section's Content slot.

   This creates a frame that defines what appears in each report row. The frame is a child of the report section component.

   a Right-click the new frame (ssGrp1) and select Slot Information. On the Single Structure Reference window, click Subclass. Click Close.

   b Right-click the same frame and select Rename. Rename it ssGrpStage.

17 Drag and drop the ssFrm component onto the group section's Content slot.

   This creates a frame that defines what appears in each report row. The frame is a child of the report section component.

   a Right-click the new frame (ssFrm) and select Slot Information. On the Single Structure Reference window, click Subclass. Click Close.

   b Right-click the same frame and select Rename. Rename it ssFrmOpportunityContent.

18 Right-click the group section component (ssGrpStage) and click Properties. In the Key property, enter ssSales_Stage. Close the Component Editor window.

19 Drag and drop the ssFrm component from the Library Browser window onto the group section's PageHeader slot.

   a Right-click the new frame (ssFrm) and select Slot Information. On the Single Structure Reference window, click Subclass. Click Close.

   b Right-click the same frame and select Rename. Rename it ssFrmStageHeader.

20 Close the Library Browser window.

When making changes in Siebel Tools for a report originally designed in Actuate e.Report Designer Professional and regenerating an ROL file, be aware of the following:

- Make sure that the ROD file that uses the ROL file is not open in Actuate e.Report Designer Professional.
- If any report fields are deleted, make sure to remove the fields from your Actuate report as well.
- Changes made to the ROL file are valid only for that instance of the ROL file. The changes will need to be entered again. However, if the changes are locally subclassed, they are safely stored in the ROD file.

Adding Label and Data Elements to the Design

Now you add data controls in the content frame (for each report row) and column heading labels in the page header (for each page break or sort break). You also define a sales stage control in the page header to display the sales stage with each group or page break.

To add text and label elements

1. Click the Library Browser button and select sscustom.rol in the Choose Included Module dialog box.
2. Drag and drop the ssTxt component from the Library Browser window onto the empty Content child slot of the content frame (ssFrmOpportunityContent).

   The Component Properties dialog box appears.
3. In the Component Properties dialog box, select the ssName field from the datastream by clicking it in the drop-down list, then click OK to dismiss the Component Properties window.
4. Resize the new data control on the layout grid to the appropriate size for opportunity name data (if it is too narrow, the name will be truncated).
5. Right-click the data control and select Rename from the pop-up menu, then give the text control a unique descriptive name (in this case, txtOpportunity).
6. Right-click the data control and select Properties from the pop-up menu, then check the CanGrow property to verify that it is set to TRUE.

   When this property is TRUE, multiple-line values in the business component fields print as multiple-line values in the report. When it is FALSE, only the first line of each value prints.
7. Repeat Step 2 through Step 6 to add the following:

   - A control called txtAccount based on ssAccount
   - A control called txtCloseDate based on ssClose_Date

   Note that you need to reposition each data control in the layout pane to the right after it is created.
8. Drag and drop the ssCur component from the Library Browser window onto the content frame ssFrmOpportunityContent.

   The Component Properties dialog box appears.
9. In the Component Properties dialog box, select the ssRevenue field from the datastream.

10 Close the Component Properties dialog box, resize and reposition the currency control on the layout grid, and rename it curRevenue.

A currency data control is used rather than a text data control when monetary values are to be displayed. This causes correct display and alignment.

NOTE: The approach described here for currency display is adequate for a test report. However, Siebel reports in version 7.0 will use txtCurrency custom control from sscustom.rol to make calculated currency values localizable (formatted according to user locale). If currency does not require calculation then the currency formatted field from the data supply library must be used in the regular ssTxt control instead.

11 Enlarge the page header frame (ssFrmStageHeader) vertically by dragging one of its handles so that it is about twice its original height.

Additional space will be required for a divider line and the sales stage data control.

12 Drag and drop the LineControl library component onto the page header frame (ssFrmStageHeader), and rename it LineSeparator.

13 Drag and drop the ssTxtSectionHeadM library component onto the page header slot, and in the Component Editor window, specify ssSales_Stage in the ValueExp property.

This text control is for display of the sales stage in the page header. It has a larger, bold font and is maroon.

14 Subclass the new data control component and rename it ssTxtStageName, then widen the control in the layout pane so that it is large enough to hold sales stage names.

15 Drag and drop the ssFrmBlueBack library component onto the page header slot.

It subclasses automatically.

16 Rename this frame component ssFrmOpportunityHeader and expand it to reveal the child slot.

This frame will contain the column headings.

17 Drag and drop the ssLblHead component onto the page header frame ssFrmOpportunityHeader, once each to create the labels listed below.

18 Set the Text property for each label as indicated.

You will need to resize, reposition, and rename each label after it is created. Each label should align vertically with the corresponding data control:

- A label called lblOpportunity, with a Text property value of Opportunity
- A label called lblAccount, with a Text property value of Account
- A label called lblCloseDate, with a Text property value of Close Date
- A label called lblRevenue, with a Text property value of Revenue

19 Drag and drop the LineControl component onto the ssFrmStageHeader frame and rename it LineSeparator.

20 Reposition the line in the layout pane so that it is above the sales stage data control.

The separator line creates a demarcation between report rows.

21 Right-click the top-level report component and choose Properties, then enter a value of Group Test Report in the ssReportTitle property.

Compiling and Testing the Report

The report is built, compiled, and run in Actuate e.Report Designer Professional. Then it can be debugged locally in Actuate e.Report Designer Professional; when it is ready to be deployed, the executable version of the report is moved to the folder where Siebel applications obtain their report executables.

To compile and install the report

1 Open Siebel Sales.

2 From the application-level menu, choose View > Site Map > Opportunities > My Opportunities.

Make sure that the desired query is active.

3 In Actuate e.Report Designer Professional, click the Run button.

The report displays in the browser window.

4 Make corrections to the report design as necessary and recompile as often as required.

5 Use Windows Explorer to copy the Grouprpt.rox file from \Siebdev\RPTSRC\enu (or equivalent) to \Siebeldev\REPORTS\ENU (or equivalent).

If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

6 In Siebel Sales, click the Reports button and from the drop-down menu select Sales Stage - Test.

The report should appear in the browser window. If there are problems, check the configuration of the report and view report object definitions in Siebel Tools.

Adding Group Totals to Reports

Group totals can appear beneath one or more numeric report columns, at the end of each group section. They provide subtotals within each group for monetary and quantity fields. A grand total of these fields can appear at the end of the report.

Group total data controls are configured in the child After frame of the group section. A group total control sums the values of a datastream variable in all the report rows in the group. To accomplish this, the Sum function is used. The scope of a Sum function within the After frame of a group section is limited to the records in that group. For additional information on the Sum function, see the information on frames and controls in *Developing Advanced e.Reports*.

This example uses the report created in "Example—Creating a Report with a Group Break" on page 81. A group total control is added to the After section of the ssGroupStage group section. It enables totaling of the revenue values for each group.

To add revenue totals to the group section

1. Click the Library Browser button, and in the Choose Included Module dialog box, select sscustom.rol.
2. Drag and drop the ssFrm component from the Library Browser window onto the group section's child After section. Subclass the frame following the instructions from in "Creating a Report Design in Actuate e.Report Designer Professional" and rename it ssFrmGroupTotals.
3. Enlarge the frame vertically in the layout pane to about twice its original height.
4. Drag and drop the ssCur component from the Library Browser window onto the ssFrmGroupTotals frame. Specify the following as the ValueExp:

   ```
   Sum(Val([ssRevenue]))
   ```

   NOTE: The above expression works only in ENU locale. To make it locale compliant tcCurrency control must be used.

5. Rename the new ssCur component curGroupRevenueTot. Reposition it in the layout frame so that it is aligned with the revenue column of the report.
6. Drag and drop the LineControl component from the Library Browser window onto the ssFrmGroupTotals frame and rename it LineTotal, then reposition it vertically in the layout pane so that it is above the revenue total control and narrow the line (using the left handle) so that it occupies only the space above the total control.
7. Save the report design, recompile, and test.

To add a final total to the report

1. Drag and drop the ssFrm component from the Library Browser window onto the child After slot of the report section (ssReportOpportunity), then subclass it following the instructions in "Creating a Report Design in Actuate e.Report Designer Professional" and rename it ssFrmReportTotals.
2. Enlarge the frame vertically in the layout pane to about twice its original height.
3. Drag and drop the ssCur component from the Library Browser window onto the ssFrmReportTotals frame and specify the following as the ValueExp:

   ```
   Sum(Val([ssRevenue]))
   ```

   NOTE: The above expression works only in ENU locale. To make it locale compliant tcCurrency control must be used.

4. Rename the new ssCur component curReportRevenueTot, then reposition it in the layout frame so that it is aligned with the revenue column of the report.

5. Drag and drop the LineControl component from the Library Browser window onto the ssFrmReportTotals frame and rename it LineRptTotal, then reposition it vertically in the layout pane so that it is above the revenue total control and narrow the line (using the left handle) so that it occupies only the space above the total control.
6. Drag and drop the ssLblb component from the Library Browser window onto the ssFrmReportTotals frame and rename it lblRptTotal.
7. Reposition it to the immediate left of the revenue total control and set its Text property to Grand Total.
8. Save the report design, recompile, and test.

   NOTE: An ssCur component assumes that all amounts are in local currency, as set in the Regional Settings in the Windows Control Panel. See "How to Display Revenue Information in Siebel Reports" on page 225 for more information.

8 Master-Detail Reports

This chapter explains how to create master-detail reports. A master-detail report displays a list of detail business component records for each record in a master business component, to which the master and detail business components have a one-to-many relationship. It is similar to a master-detail view in a Siebel application, in that detail records are displayed for each master record. Unlike a master-detail view, a master-detail report lists detail records for all master records at once, rather than for one master record at a time.

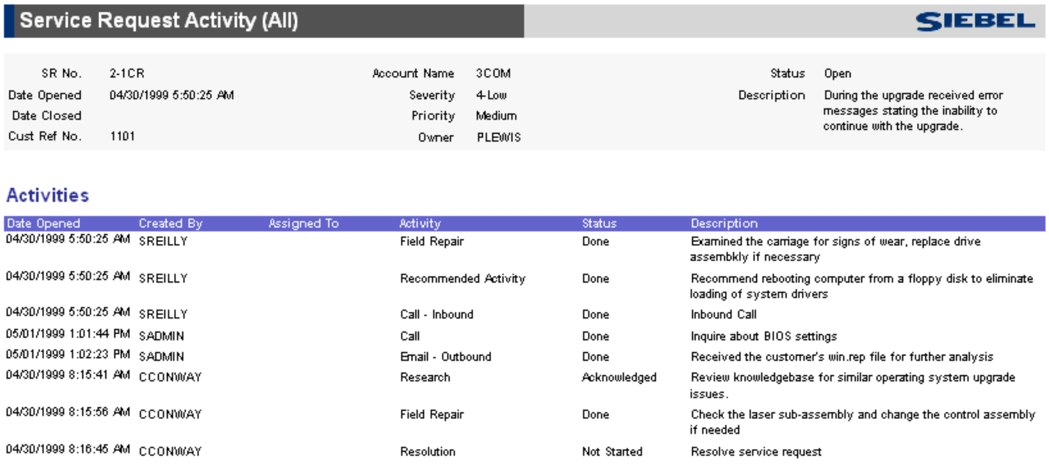
A report with record lists for a detail business component is said to contain a *subreport*. In some ways this term is misleading, because it implies the presence of one list of detail records. In fact, there is one list of detail records (one subreport) following each master record. However, in both Siebel Tools and the report design, a single subreport is specified, with the result that one list of detail records appears for each master.

This chapter consists of the following topics:

About Master-Detail Reports

An example of a master-detail report is the Service Request Activity (All) report in Siebel Service, shown in Figure 13.

This report provides master information for each service request, followed by the list of activities for that service request. Each service request begins on its own page. This report is analyzed in "How Master-Detail Reports Work" on page 96.



Service Request Activity (All)

SIEBEL

SR No.	2-1CR	Account Name	3COM	Status	Open
Date Opened	04/30/1999 5:50:25 AM	Severity	4-Low	Description	During the upgrade received error messages stating the inability to continue with the upgrade.
Date Closed		Priority	Medium		
Cust Ref No.	1101	Owner	PLEWIS		

Activities

Date Opened	Created By	Assigned To	Activity	Status	Description
04/30/1999 5:50:25 AM	SREILLY		Field Repair	Done	Examined the carriage for signs of wear, replace drive assembkly if necessary
04/30/1999 5:50:25 AM	SREILLY		Recommended Activity	Done	Recommend rebooting computer from a floppy disk to eliminate loading of system drivers
04/30/1999 5:50:25 AM	SREILLY		Call - Inbound	Done	Inbound Call
05/01/1999 1:01:44 PM	SADMIN		Call	Done	Inquire about BIOS settings
05/01/1999 1:02:23 PM	SADMIN		Email - Outbound	Done	Received the customer's win.rep file for further analysis
04/30/1999 8:15:41 AM	CCONWAY		Research	Acknowledged	Review knowledgebase for similar operating system upgrade issues.
04/30/1999 8:15:56 AM	CCONWAY		Field Repair	Done	Check the laser sub-assembly and change the control assembly if needed
04/30/1999 8:16:45 AM	CCONWAY		Resolution	Not Started	Resolve service request

Figure 13. Service Request Activity (All) Report

A master-detail report can also have multiple subreports. In this case, a list of detail records appears for each of a number of business components for each master record. For example, the Account Service Profile report provides three lists for each account master record: customer survey responses, opportunities, and service requests. A report with two subreports is described in "Example—Creating a Master-Detail Report" on page 100. Master-detail reports with multiple subreports are common among the Siebel standard reports.

When starting a master-detail report, no need exists to start the report from a view where all business components used in the report are present. Having the master business component in the view of the report is enough for the report to retrieve all the child or grandchild records.

The only requirement is that all business components used in the report should have been added to the business object used by the view from where the report is started. Also, the business components should use the correct links to establish the proper relationships among them, as an Actuate report retrieves the data using the Siebel Business Object Layer.

If a Multi Value Field (MVF) is included in a report, only the first record is displayed. In order to display all the records from an MVF in the report, a subreport should be created in Tools under the associated Report object. The subreport should be based on the business component that contains the MVF to be displayed. Make sure that this business component is included in the business object pertaining to the Report object. If the business component is not already included in the business object, it should be included after defining an appropriate link.

This information is also valid for indirect MVFs. For example, consider the case where the business address (an MVF) of an Account associated with an Opportunity should be displayed in the report. The business addresses in the MVF are not directly related to the Opportunity, but they are related to the Account that it is associated with it. To display all the records in the business address MVF as a subreport, first create a link between Business Address business component and Opportunity business component using Account Id as the source field. Then include Business Address business component under Opportunity business object. Create a subreport with Business Address business component under the Report object and include the necessary MVF to display.

How Master-Detail Reports Work

Figure 14 illustrates the structure of a master-detail report in Actuate.

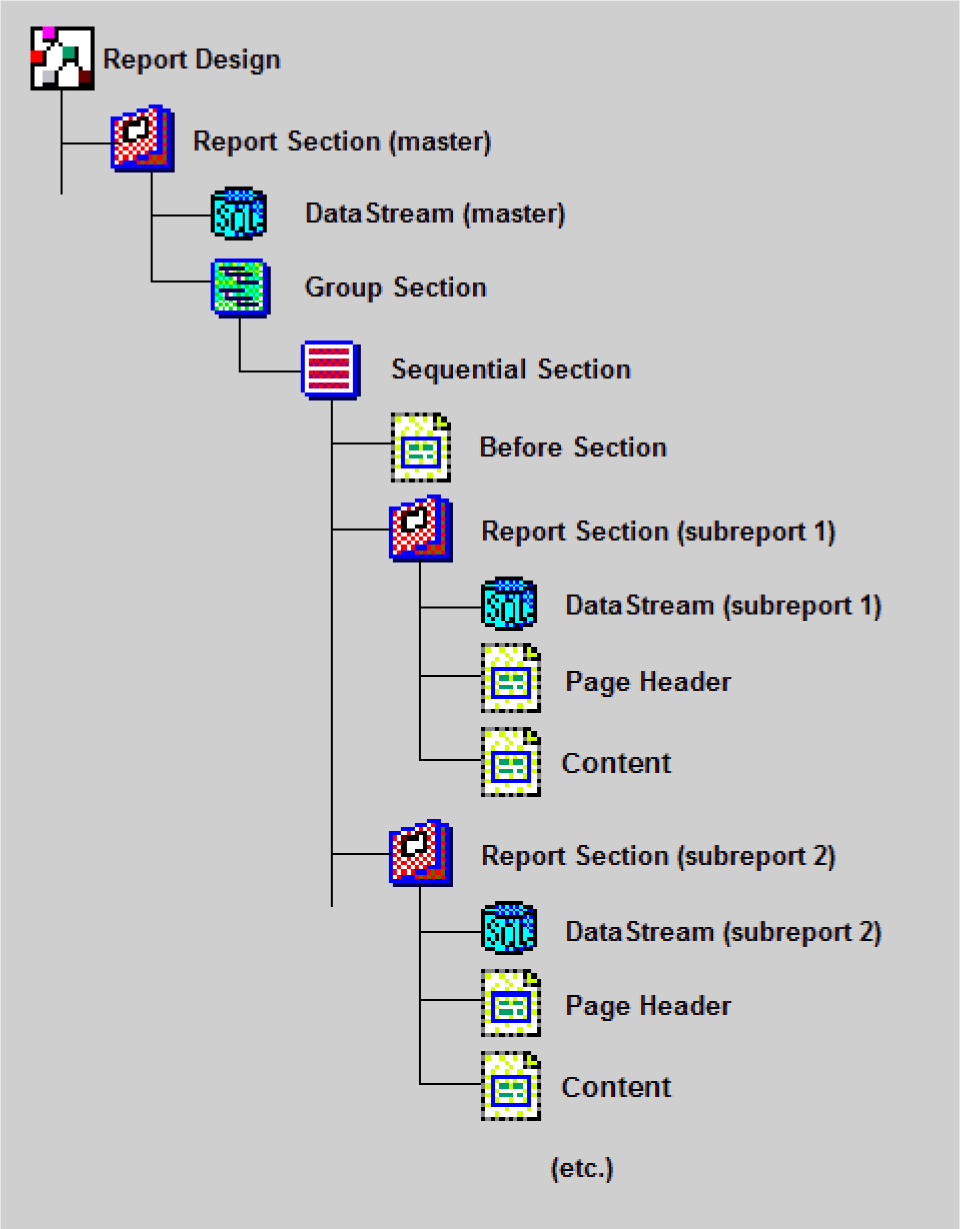


Figure 14. Structure of a Master-Detail Report

This includes the following major components:

- **Report Design.** This is the top-level component in a report and corresponds to the ROD file in which it resides. There are no special features of the report design component for master-detail reports.
- **Master Report Section.** The master report section defines the data acquisition for and display of each master record and, through its child components, defines the subreports.
- **Master Datastream.** The master datastream component defines the source of data for the master records. A separate datastream is defined in each subreport to obtain the detail data. As in group reports, the data must be sorted with the group field as the primary sort key.
- **Group Section.** The group section component implements grouping and group break behavior for the master records in its parent report section. This behavior is defined in the group section's Key property and other properties, such as PageBreakBefore. The subreports are children of the group section component.
- **Sequential Section.** A sequential section causes its child section components to execute one after another in sequence. In a master-detail report, this causes the Before frame to execute before the first subreport section, followed by each additional subreport section if any are present.
- **Before Section.** This section holds the frame that appears once for each master record, displaying master information before the subreports.
- **Subreport Report Section.** Each subreport section defines one subreport in the master-detail report. It has its own datastream and content frames, defined with child components.
- **Subreport Datastream.** Each detail datastream provides data to the detail records in one subreport (its parent report section). The data is obtained through a subquery on the master business component query, requesting all detail records for the current master record.
- **Subreport Page Header Frame.** In a subreport report section, the page header frame defines the heading information for the subreport records. This normally includes a title identifying the subreport and column heading labels to appear above the subreport records.
- **Subreport Content Frame.** The content frame defines the layout of one subreport row.

To learn more about the configuration of a simple master-detail report, it is helpful to open a standard Siebel report of this type. A good report to study for this purpose is the Service Request Activity (All) report, which is invoked from a filter in the Service screen in Siebel Service (typically the My Service Requests filter). You should examine the report output in Siebel Service, and the report design in Actuate e.Report Designer Professional, leaving both open to compare them.

To generate the Service Request Activity (All) report in Siebel Service

1. Open Siebel Service.
2. Click Site Map and navigate to Service > Service Request List.

   My Service Requests is the default view.
3. Click the Reports button, and from the drop-down list select Service Request Activity (All).

The Service Request Activity (All) report appears in the browser window, as shown in Figure 13 on page 94.

To open the report design for the Service Request Activity (All) report

1. Open Actuate e.Report Designer Professional.
2. Choose File > Open.
3. In the Open dialog box, navigate to the C:\Siebdev\RPTSRC\ENU\STANDARD folder (or the equivalent on your computer) and choose srvreqaa.rod.

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

Figure 15 shows the srvreqaa.rod report design file in Actuate e.Report Designer Professional.

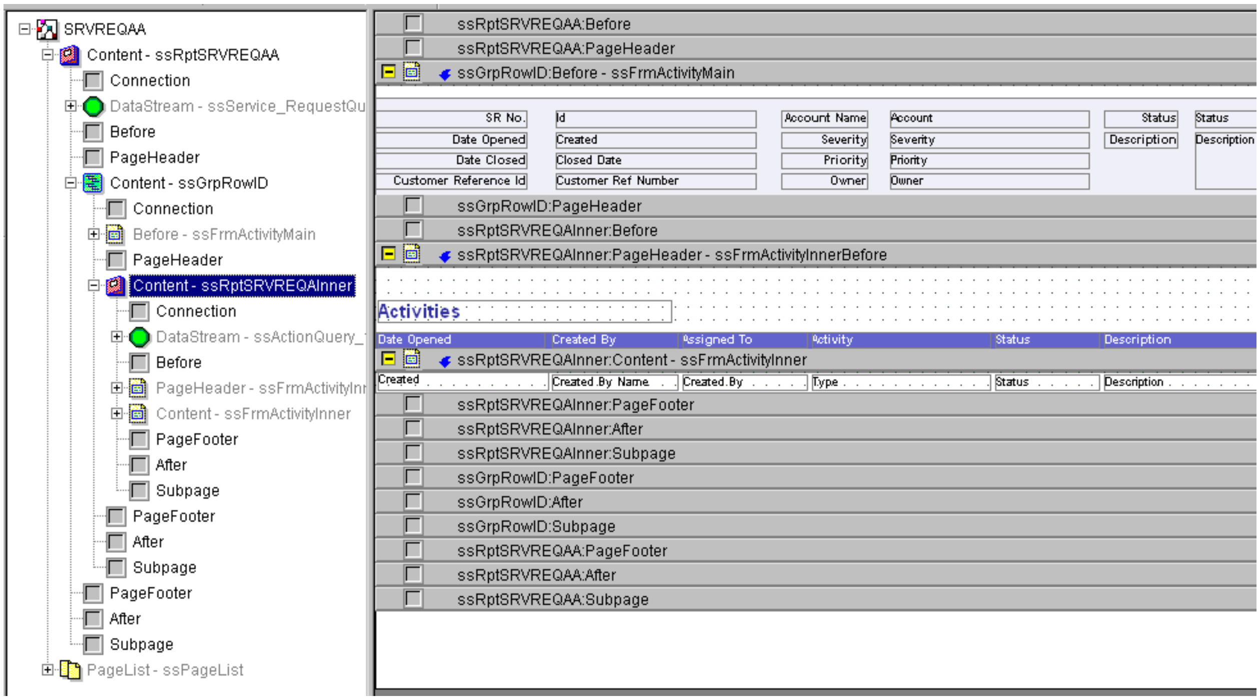


Figure 15. Service Request Activity (All) Report Design

Explore the report design, compare design elements to the corresponding features in the report output, and right-click components to view their property lists.

Notice the following features of this report design:

- There is a page break before each new service request (master) record. This is configured with a value of TRUE for the PageBreakBefore property of the group section component (ssGrpRowID).

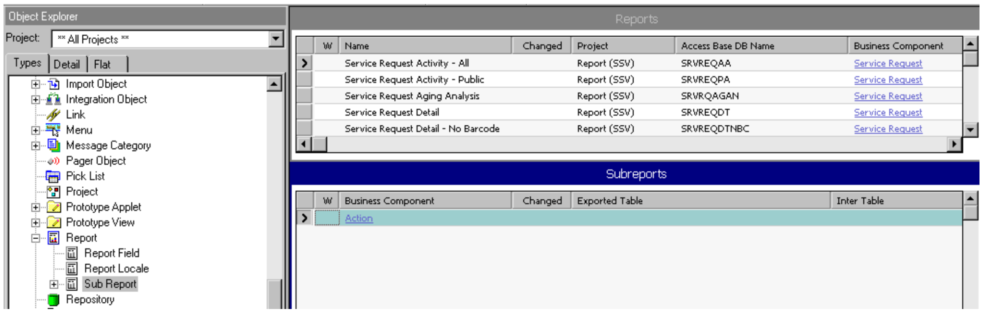
- The Before frame in the group section, and the page header and content frames in the report section, are published components obtained from the ssSrvReq.rol custom component library. The contents of these frames are used identically in the two reports that list activities by service request.
- The two datastreams, one (ssService_RequestQuery) in the master report and one (ssActionQuery_1) in the activity subreport, are obtained from the data supply library file, srvreqaa.rol. In the data supply library file for a master-detail report, one datastream is provided for the master report and one for each subreport.

Additional useful information is obtained by viewing the report object definition (and children) for this report in Siebel Tools.

To view the report object definition and children in Siebel Tools

1. Open Siebel Tools.
2. Navigate to the Report object type in the Object Explorer and expand it.

   Note the two child object types, Report Field and Sub Report.
3. In the Reports window in the Object List Editor, navigate to the Service Request Activity - All object definition.
4. Click on the Sub Report object type in the Object Explorer. A second Object List Editor window opens, displaying subreport child object definitions of the current parent report, as shown in the following figure.



Project: ** All Projects **

Reports

W	Name	Changed	Project	Access Base DB Name	Business Component
	Service Request Activity - All		Report (SSV)	SRVREQAA	Service Request
	Service Request Activity - Public		Report (SSV)	SRVREQPA	Service Request
	Service Request Aging Analysis		Report (SSV)	SRVRQAGAN	Service Request
	Service Request Detail		Report (SSV)	SRVREQDT	Service Request
	Service Request Detail - No Barcode		Report (SSV)	SRVREQDTNBC	Service Request

Subreports

W	Business Component	Changed	Exported Table	Inter Table
	Action			

   The Service Request Activity - All report has one child subreport object definition: Action. The Action subreport object definition defines the activity subreport.

   Notice that the business component property setting for the Action subreport is Action. There is no separate report name, because a subreport in Actuate is internal to the report design that uses the subreport object definition's parent.

5. Expand the Sub Report object type and select Sub Report Field. Notice the list of fields defined as children of the Action subreport object definition.

   Subreport field object definitions perform the same role for a subreport as report field object definitions do for a report—namely, defining fields to export to the report or subreport from the specified business component.

Example—Creating a Master-Detail Report

In this example, an account report is created, using Actuate e.Report Designer Professional, that provides an opportunity list and a contact list for each account. The following tasks are required to create the report:

- Create and export a new report object definition (and children) in Siebel Tools.
- Create a custom component library.
- Create a report design in Actuate e.Report Designer Professional.
- Add datastreams to the report design.
- Add frame, data control, and label elements to the design.
- Compile and test the report.

Creating a New Report Object Definition in Siebel Tools

Each report design normally has its own report object definition in Siebel Tools and a corresponding data supply library file. In Chapter 7, "Reports with Group Sections," the report object definition was copied from an existing one, and minor changes were made. In this example, the same technique is employed to save time, but more changes are made.

To create a new report object definition in Siebel Tools by copying

1. Open Siebel Tools. Verify that the Account business object has Opportunity and Contact detail business components relative to the Account (master) business component.

   This is an unnecessary step in this example, but one you should employ as a precaution when using less common business component relationships. The master-detail relationships specified by the business object components and links in the business object are necessary for any master-detail datastream relationship to work.

   The hierarchy of the reports and subreports must be in accordance with the configuration previously done (business object and its business component with the proper links).

   NOTE: The correct approach is to use only the driving business component in a view as the main datastream in the report, because problems may arise otherwise.

2. Navigate to the Report object type in the Object Explorer. Expand this object type to reveal the child Sub Report object type.

3. Locate and select the Account Summary object definition in the Object List Editor window for reports.

   Notice that this report has seven subreport children, including Opportunity and Contact. Any account report object definition with these latter two subreports would have sufficed. You will copy this report object definition and delete the unwanted child subreports.

   a. Lock the Report project (choose Repository >Lock Project).

b. With the Account Summary object definition selected, choose Edit > Copy Record.

   A new report object definition is created with empty Name and Menu Text properties and a set of children and properties otherwise matching the original.

4. Change the following property settings in the new report object definition (the other property settings do not need to be changed):

   - **Name.** Account - Opty/Contact Detail

     This is the name used to refer to the report structure in Siebel Tools, for instance when you add the report to the Reports menu for a view.

   - **Access Base DB Name.** ACOPCOM

     This specifies that Acopcom.rox will be the name of the report executable invoked when this report object definition is invoked from the Reports menu in a view.

   - **Menu Text.** Account - Opportunity/Contact Detail

     This is the text that will appear in the Reports menu for a view when this report is included in the view through the use of a view report object definition.

   - **Template Name.** ACOPCOM

     Specifies the name to be used for the generated data supply library file when the Generate Actuate Report option is invoked for this report object definition.

   - **Sort Specification.** (blank)

     By default, sorts the account records into account name order, which is necessary for the group breaks to work correctly. Alternatively, you could specify a value of Account.

5. Navigate to the Sub Report list in the Object List Editor for the new report object definition.

6. Delete all the subreport object definitions, using Edit > Delete Record, except Opportunity and Contact.

7. Expand the Sub Report Fields object definition for the Opportunity subreport.

8. Add the following subreport fields if they are not already present: City, Close Date, Description, Name, Postal Code, Rep%, Revenue, Sales Rep, and Sales Stage.

9. Export to a data supply library file by choosing Tools > Generate Actuate Report.

   This generates a data supply ROL file, named according to the value in the Template Name property, in the C:\Siebdev\RPTSRC\ENU\LIB folder (or equivalent on your system).

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

10. Unlock the Report project and navigate to the View object type in the Object Explorer.

a In the Object List Editor, locate the Account List View object definition. In the Object Explorer, expand View and click View Report.

 This is the My Accounts view in the Accounts screen.

b Add a View Report object definition in the Object List Editor.

 Enter the name “Account - Opty/Contact Detail” (match spelling and spacing exactly).

c Add a sequence number that is greater by 1 than any other sequence number in the list.

11 Choose Repository > Compile and recompile the repository. Move the resulting SRF file to C:\Siebel\objects (or the equivalent location on your system).

Creating a Custom Component Library

In this exercise, a new report design is created using library components. Components from other report designs and custom libraries are used where possible to shorten the design time. Custom component libraries are described in “Using a Custom Component Library” on page 69. A custom library already exists with opportunity detail frame information: ssOppt.rol. However, the account master and contact detail frame layouts need to be published to a new custom component library from an existing report design, in this case srvreqaa.rod.

To create a custom component library with account master and contact detail frames

1. Open the srvreqaa.rod (Service Request Activity - All) report design in Actuate e.Report Designer Professional.
2. Choose Tools > Library Organizer > New Library. Specify the name sssrvreq.rol and the path C:\Siebdev\RPTSRC\ENU\LIB.

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.
3. Drag and drop the following frames from the srvreqaa report design into the sssrvreq.rol custom library (choose the Publish The Component option for each in the Drop Component dialog box):
   - ssFrmHeadSection. This is the Before frame for the group section, and displays account header information.
   - ssFrmContactsBefore.
   - ssFrmContactsContent.
4. Close the srvreqaa.rod report design and click the Save All button when prompted to save the report design and library.

The srvreqaa.rod report and your new Account - Opportunity/Contact Detail report will share frame components from this library. If any customizations need to take place in either report, the component is subclassed first to keep the changes local to one report.

Creating a Report Design in Actuate e.Report Designer Professional

Now the new report design is created and the major structural components are added.

To create a master-detail report design in Actuate e.Report Designer Professional

1. Open Actuate e.Report Designer Professional.
2. Choose File > New. In the Create New Report window, select Blank Report and click OK.

   A blank design report appears.

   NOTE: Another option for creating a master-detail report design can be found in the postinstallation tasks for Actuate e.Report Designer section in *Installation Guide* for the operating system you are using.

3. Delete the following components from the structure tree.

   NOTE: You will need to expand the Content - Report section on the structure pane to see the components.

   - **Connection component.** This is a predefined connection that is not used to create Siebel reports.
   - **DataStream component.** This component will be replaced with the datastream created in Siebel Tools.
   - **Content - Frame component.** This component will be replaced with a subclassed frame.

4. Choose Tools > Library Organizer and click More.

   The Include Library window appears.

   a. Click Browse and navigate to C:\Siebdev\RPTSRC\ENU\LIB (or equivalent), choose sssiebel.rol, and click Open. Click OK to close the Include Library window.

      If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

      See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

   b. Repeat this step for the files sscustom.rol, sssrvreq.rol, and Acopcon.rol.

5. Enter the following information for the report:

   - **Title.** Account - Opty/Contact Detail

- **Filename.** C:\Siebdev\RPTSRC\ENU\Acopcon.rod

  This is the filename and folder location where the report design file you are creating will be saved. If you have a \custom subfolder of ...\ENU, save it there instead.

  If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

  See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

- **Report Root Name.** Acopcon

  This specifies the name of the top-level report design object in the design file.

6 Choose File > Save. Specify Acopcon.rod in the ...\enu (or ...\enu\custom) folder.

If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \enu folder. Instead, you have a folder in the appropriate language code for your installation, such as \deu for Germany.

See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

Remember to save the design file periodically as you work.

7 Double-click the Acopcon (top-level) icon in the structure tree (or right-click it and select Properties).

a In the Component Editor window, click the Class tab.

b Change the Super Class value from AcReportSection to ssRpt.

This is an important step. Siebel reports must inherit from ssRpt (the Siebel standard report) instead of directly from AcReportSection (Actuate report). ssRpt is provided in sscustom.rol and inherits from baseReport from sssiebel.rol.

NOTE: You cannot perform this step until you have included the sssiebel.rol library.

Two components are automatically added to the report when you change the report's class: a report section (ssRpt) and a pagelist section (ssPageList).

8 Make sure that the Show Empty Slots option in View >Options is checked.

9 Expand the top-level report component.

a Right-click the report section child component in the structure tree (Report) and select Slot Information.

The Single Structure Reference window appears.

b Change Name in the Contents section to ssRpt and click Close.

c Right-click the report section component (ssRpt1) and click Subclass.

d Right-click the report section (ssRpt1) and click Rename. Change the name to ssRptAccount.

10 Expand the report section component (ssRptAccount).

   a. Click the library button and click sscustom.rol.
   b. Drag and drop the ssGrp library component onto the Content child slot of the report section.

11. Right-click the group section component (ssGrp) and subclass following Step 9 on page 104.
   a. Right-click the group section component (ssGrp1) and click Rename.
   b. Change the name to ssGrpAccount.
   c. Right-click the group section component (ssGrpAccount) and click Properties. In the Key property, enter ssName.
   d. Close the Component Editor window.

12. Drag and drop the ssSeq library component from the Library Browser window to the content section (ssGrpAccount). When prompted, click Use As Content.

    This provides the sequential section that causes the subreports to execute in sequence.

13. Subclass the sequential section and rename it ssSeqAccount.
   a. Drag and drop the ssRpt library component onto the sequential section. Subclass following Step 9 on page 104 and rename it ssRptOpportunity. Expand this report component.
   b. Drag and drop the ssRpt library component onto the sequential section again. Subclass this report component following Step 9 on page 104 and rename it ssRptContact. Expand this report component.

14. Close the Library Browser window.

Adding Datastreams to the Report Design

Three datastreams are added to report sections in the report design from the data supply library, one each for the master report and the two subreports.

To add datastreams to the report design

1. Click the Library Browser button and select Acopcon.rol.
2. Drag and drop the ssAccountQuery library component onto the DataStream slot beneath the master report section, ssRptAccount.
3. Drag and drop the ssOpportunityQuery_1 library component onto the DataStream slot beneath the opportunity detail report section, ssRptOpportunity.
4. Drag and drop the ssContactQuery_2 library component onto the DataStream slot beneath the contact detail report section, ssRptContact.
5. Close the Library Browser window.

Adding Frame, Data Control, and Label Elements to the Design

Many of these elements will be obtained from two custom component libraries, one of which you have already created.

To add frame, data control, and label elements

1. Click the Library Browser button and in the Choose Included Module dialog box, select sssrvreq.rol.
2. Drag and drop the ssFrmHeadSection library component onto the Before child slot of the group section, ssGrpAccount.
3. Collapse the component subtree for ssFrmHeadSection by clicking its minus-sign icon.

   Component subtrees for headers and detail rows are distracting in expanded format and should be collapsed for visual clarity and ease of navigation.
4. Drag and drop the ssFrmContactBefore library component onto the page header child slot of the contact subreport section, ssRptContact, then collapse the component subtree for ssFrmContactBefore.
5. Close the Library Browser window, and choose Tools > Library Organizer.

   Select ssOppt.rol in the ...\enu\library folder.

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \enu folder. Instead, you have a folder in the appropriate language code for your installation, such as \deu for Germany.

   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

   This custom component library is used in opportunity reports; it has the page header and detail line needed for the opportunity subreport you are creating.
6. Drag and drop the ssOppHeader library component onto the page header child slot of the opportunity subreport section, ssRptOpportunity, then collapse the component subtree for ssOppHeader.
7. Drag and drop the ssOppRowNoAddress library component onto the Content child slot of the opportunity subreport section, ssRptOpportunity, then collapse the component subtree for ssOppRowNoAddress.

   Close the Library Browser window.
8. Locate and expand the page header child frame of the ssRptOpportunity report section, then subclass this page header frame (ssOppHeader) and rename it ssFrmOpportunityPage.

   You need to make some modifications to this page header.
9. Select the ssTxtOPPTHeadM child data control of ssFrmOpportunityPage and delete it.
10. Open the sscustom.rol library and drag and drop the ssLblSectionHead component from the Library Browser window onto the ssFrmOpportunityPage page header.

11 Rename the new section head label ssLblOpportunityHead, then resize and reposition the header label so that it is comparable to the one for contacts.

Close the Library Browser window.

12 Select the ssFrmOPSTATEHeader content frame in ssFrmOpportunityPage, then subclass it and rename it ssFrmOpportunityHeader.

13 Expand ssFrmOpportunityHeader to show its child label components.

You will delete the account label from this header and the detail data row that appears beneath it because the account is already displayed in the master report.

14 Select the ssLblAccount label child of ssFrmOpportunityHeader and delete it.

15 Subclass each of the remaining label children of ssFrmOpportunityHeader and drag them to the left to fill up the space emptied by deleting ssLblAccount.

By clicking each label in the opportunity header while holding down SHIFT, you can create a grouping of labels that you can drag to the left together, maintaining the distances between them and saving time.

16 Collapse the ssFrmOpportunityPage component tree.

a Expand the ssOpRowNoAddress content frame child of ssRptOpportunity.

b Subclass ssOpRowNoAddress and rename it ssOpportunityContent.

c You will now delete the account data control and reposition the other data controls to fill the empty space.

17 Select the ssTxtOPSTATEAccount data control and delete it.

18 Subclass each of the remaining data control children of ssFrmOpportunityRow and SHIFT-click each of them in the layout pane (except the name control). Drag them to the left to fill up the space emptied by deleting ssTxtOPSTATEAccount.

19 Make sure that the data controls line up with the header labels above them.

20 Make sure that the PageBreakBefore property of the group section (ssGrpAccount) is TRUE and that its PageBreakAfter property is FALSE. Also make sure that the PageBreakBefore and PageBreakAfter properties of the ssRptContact and ssRptOpportunity subreport sections are all set to FALSE.

If you wanted the subreport sections on their own pages, you would set PageBreakBefore to TRUE in the respective subreport sections. With the FALSE settings in all four subreport page break properties, there are no page breaks except between accounts.

21 Open the Properties window for the top-level report design component, Acopcon, and set the ssReportTitle property to Account - Opportunities/Contacts Detail.

Compiling and Testing the Report

The report is built, compiled, and run in Actuate e.Report Designer Professional. Debug the report locally in Actuate e.Report Designer Professional and, when it is ready to be deployed, the executable version of the report is moved to the folder where Siebel applications obtain their report executables.

To compile and install the report

1. Open Siebel Sales.
2. From the application-level menu, choose View > Site Map > Accounts > My Accounts.

   Make sure that the desired query is active.
3. In Actuate e.Report Designer Professional, click the Run button.

   The report appears in the browser window.
4. Make corrections as necessary to the report design and recompile as often as required.
5. Use Windows Explorer to copy the Acopcon.rox file from C:\Siebdev\RPTSRC\enu (or equivalent) to C:\Siebel\reports\enu (or equivalent).

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \enu folder. Instead, you have a folder in the appropriate language code for your installation, such as \deu for Germany.

   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.
6. In Siebel Sales, click the Reports button and from the drop-down list, select Account - Opportunity/Contact Detail.

   The report should appear in the browser window. If there are problems, check the configuration of report and view report object definitions in Siebel Tools.

Example—Creating a Report with Multiple Hierarchies

In this example, an account report is created that provides a list of opportunities and associated competitors for each opportunity, and a list of contacts for each account. Notice that this report displays three levels of hierarchy (parent > child > grandchild) relationship. It is possible to create reports with multiple hierarchies using the methodology laid out in this example.

The following tasks are required to create the report

1. Create and export a new report object definition (and children and grand children) in Siebel Tools.
2. Create a custom component library.
3. Create a report design in Actuate e.Report Designer Professional.
4. Add datastreams to the report design.
5. Add frame, data control, and label elements to the design.
6. Compile and test the report.

Creating a New Report Object Definition in Siebel Tools

Only Step 1 from "Example—Creating a Report with Multiple Hierarchies" is described in detail. All the other steps for developing a Master-Detail report are identical to those explained in "Creating a New Report Object Definition in Siebel Tools" on page 100.

To create a new report object definition in Siebel Tools by copying

1. Open Siebel Tools. Verify that the Account business object has Opportunity, Contact, and Competitor detail business components relative to the Account (master) business component. Also verify under Account business object that a link is defined for the Competitor business component between Opportunity and Competitor business components.

   This confirms the hierarchical relationship among the business components under Account business object.

   In this example, the Opportunity and Contact business components are at the child level, and the Competitor business component is at the grandchild level relative to Account business object.

2. Make sure that all the business components to be used in the master detail report are active in the associated business object.

   A master detail report can be created with multiple hierarchy levels with as many business components as the user needs, as long as all of these business components belong in the same business object and are active. Further, each business component below the parent level should have a link defined with a parent level business component.

   In this example, make sure that the competitor business component (which will appear at the grandchild level in the report) has a link defined with Opportunity business component.

3. Navigate to the Report object type in the Object Explorer. Expand this object type to reveal the child Sub Report object type.

4. Locate and select the Account Summary object definition in the Object List Editor window for reports.

   Notice that this report has seven subreport components, including Opportunity, Contact, and Competitor. Any account report object definition with these latter three subreports would have sufficed. You will copy this report object definition and delete the unwanted subreport components.

5. Lock the Report project (choose Tools > Lock Project).

6. Select the Account Summary object definition and choose Edit > Copy Record.

   A new report object definition is created with empty Name and Menu Text properties and a set of children and properties otherwise matching the original.

7. Change the following property settings in the new report object definition (the other property settings do not need to be changed):

   - **Name.** Multiple Hierarchy Report

     This is the name used to refer to the report structure in Siebel Tools, for instance when you add the report to the Reports menu for a view.

- **Access Base DB Name.** MULHIER

  This specifies that Mulhier.rox will be the name of the report executable invoked when this report object definition is invoked from the Reports menu in a view.

- **Menu Text.** Multiple Hierarchy Report

  This is the text that will appear in the Reports menu for a view when this report is included in the view through the use of a view report object definition.

- **Template Name.** MULHIER

  Specifies the name to be used for the generated data supply library file when the Generate Actuate Report option is invoked for this report object definition.

- **Sort Specification.** (blank)

  By default, do not sort the account records into account name order, which is necessary for the group breaks to work correctly. Alternatively, you could specify a value of Account.

8 Navigate to the Sub Reports list in the Objects List Editor for the new report object definition.

9 Delete all the subreport object definitions (using Edit > Delete Record) except Opportunity, Contacts, and Competitor.

Notice that the position field indicated for Opportunity and Contacts is 1 and 2 respectively, while for Competitor it is 1.1. This indicates that Opportunity and Contacts are at hierarchical level 2 and Competitor is at level 3 under Opportunity.

The position is indicated relative to the main business component. Each hierarchy level is separated by a period, and the number indicates the relative position with respect to the associated parent. For example, position 2.3 indicates that the subreport is two levels below the main business component, and that it is a child, 3, of the business component designated with position 2.

10 Expand the Sub Report Fields object definition for the Opportunity subreport.

11 Add the following subreport fields if they are not already present: City, Close Date, Description, Name, Postal Code, Rep%, Revenue, Sales Rep, and Sales Stage.

12 Expand the Sub Report Fields object definition for the Competitor subreport. Add the subreport fields Threat Value, and Vendor if they are not already present.

13 Export to a data supply library file by choosing Tools > Utilities > Generate Actuate Report.

This generates a data supply ROL file, named according to the value in the Template Name property, in the C:\Siebdev\RPTSRC\ENU\LIB folder (or equivalent on your system).

If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

14 Unlock the Report project.

15 Navigate to the View object type in the Object Explorer.

- a In the Object List Editor, locate the Account List View object definition.
- b In the Object Definition, expand View and click on View Report. This is the My Accounts view in the Accounts screen.
- c Add a view report object definition in the Object List Editor.
- d Enter the name Multiple Hierarchy Report (match the spelling and spacing exactly) and a sequence number that is one higher than any other sequence number in the list.

16 Choose Tools > Compile and recompile the repository. Move the resulting SRF file to C:\Siebel\objects (or the equivalent location on your system).

Creating a Report Design in Actuate e.Report Designer Professional

In Creating a Master–Detail Report example, you first created a custom component library and then a new report design in Actuate e.Report Designer Professional. In this example, you will create a report design by first copying an existing design and making changes to the copy.

To create a report design with multiple hierarchy

1 Open the Acopcom.rod (Account – Opty/Contact Detail) report design in Actuate e.Report Designer Professional and choose File > Save As and save the report design as Mulhier.rod.

2 Highlight the Report at root level in the design editor and double-click to open the Component Editor. In the Properties tab, change the title by entering Multiple Hierarchy Report for the ssReportTitle parameter.

3 Choose Tools > Library Organizer to open the Library Organizer window.

In the Libraries Included in Your Report frame, highlight Acopcom.rol and click the down arrow. If Mulhier.rol is available in the Available Libraries frame, click the up arrow to add to the included libraries.

- a If Mulhier.rol is not in the Available Libraries frame, browse through the file system clicking More and navigate to the directory that contains Mulhier.rol library.
- b Select this library in the Available Libraries frame and click the up arrow to add it to the included libraries.
- c Click the Library Browser button and select the Mulhier.rol library to open the datastreams of Account (ssAccountQuery), Opportunity (ssOpportunityQuery_1), Contacts (ssContactsQuery_2), and Competitor (ssCompetitorQuery_1_1) business components.
- d Only include the datastream of Competitor business component.

4 Expand the ssRptOpportunity report component in the design editor.

5 Drag and drop the ssSeq library component from the Structure icon into the content section.

This provides the sequential section that causes the subreports to execute in sequence.

6 Subclass the sequential section and rename it ssSeqOpty.

7. Drag and drop the ssRpt library component on the sequential section.

   a. Subclass it and rename it ssRptCompetitor.

   b. Expand this report component.

8. Drag and drop data stream object ssCompetitorQuery_1_1 into the DataStream node of ssRptCompetitor.

9. Close the Library Browser Window.

Adding Frame, Data Control, and Label Elements to the Design

See "Adding Frame, Data Control, and Label Elements to the Design" on page 106 to add frame, data control, and label elements to the Threat Value and Vendor fields in the Competitor section (the other sections already have the controls added).

9 Composite Datastreams

This chapter consists of the following topics:

- "About Datastream Concepts" on page 113
- "Using Composite Datastreams in Reports" on page 114

About Datastream Concepts

A datastream component represents the data from one business component. Datastreams and report sections have a one-to-one relationship, and multiple datastreams are common in report designs. These relationships drive many of the techniques that are used in more complex reports.

Each datastream uses a data row component to store variable values and return formatted data in the report instance. In Siebel/Actuate, the data row component represents the list of fields in the report object definition. Each field in the report object definition maps to one variable defined in the data row. A data row object is created for each row of data in the report object definition.

The data row component is a class that is instantiated to create data row objects at run time. The datastream contains the code to extract data; the data row holds the values. Since data rows are almost wholly defined by their variable attributes, you should modify the report object definition and its children and avoid writing code to the data row component.

A data row object, however, is visible to the report section and can be intercepted after it is created and before it populates controls. You can create a local subclass of the datastream and data row components, and override the Fetch method in the datastream (to process the entire record) or the OnRead method in the data row (to process a single field).

Though a datastream component uses only one data row component in its local space (report section), the creation of a global data row can make fields from multiple data rows available to the entire report design. This technique is described in "Using Composite Datastreams in Reports" on page 114.

In Actuate, a datastream is a collection of components that deliver data to the report. Because Siebel data access is done through OLE and the OMU (the Reports Server interface), queries are always executed outside the Actuate environment.

The role of the datastream is to create an interface through OLE or the OMU and deliver the formatted data row to the report design. Siebel Tools defines the variables required to execute the queries, their values being determined by the requirements of the report design. The datastream variables, with their types and functions, are described in Table 8.

Table 8. Siebel Datastream Class Variables

Variable	Type	Function
ssAppServer	Integer	Pointer to the object created in the ssReport::Build Report () method of the datastream. It identifies the active Siebel application server or model.
ssBO	Integer	Pointer to the object created in the Start method. It identifies the active business component.
ssBusCompName	Integer	Pointers to the master and child business components used to obtain data from the view in the Siebel application.

The overridden datastream methods are covered in Appendix B, "Method Reference."

Using Composite Datastreams in Reports

A report often requires a single frame to display data from multiple business components. Each frame gets data from its report section, and each report section has one datastream. Each datastream is associated with a single business component through the report object definition. Normally, a data control cannot see beyond the bounds of its report section.

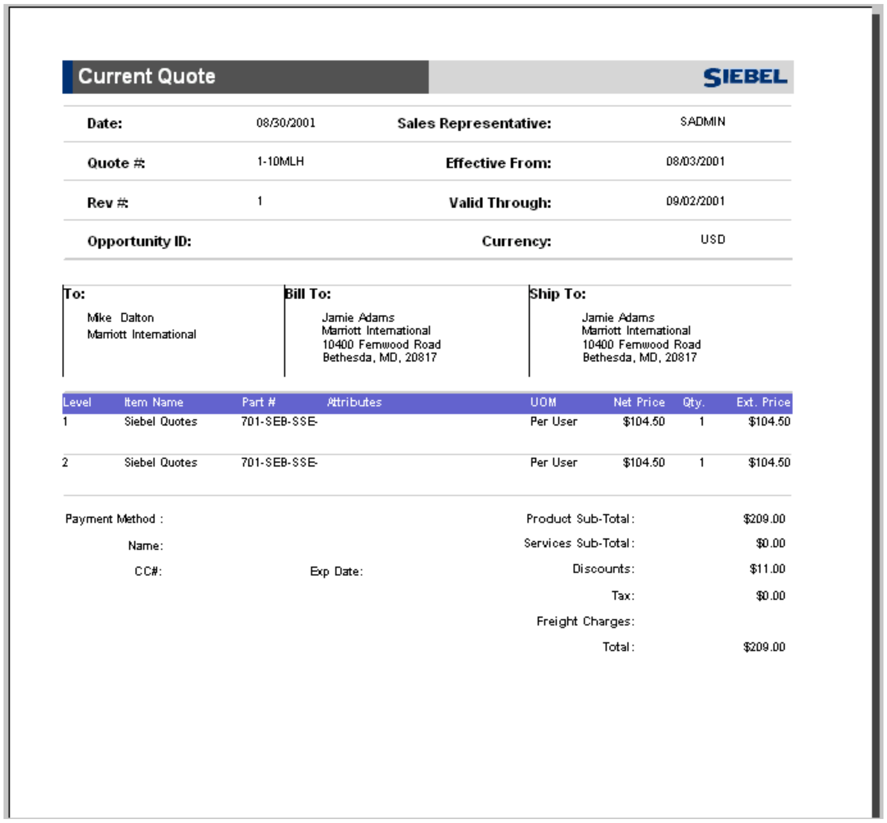
The solution is to create a data row, global in scope, that is visible to the entire report, not just to its report section. You do this by putting a variable on the report design component. It is then possible to populate the control with the proper data.

The following tasks are required to create and use a global data row:

1. Add global variables to the report design.
2. Modify the master datastream component.
3. Reference global variables in controls.

These steps are illustrated with the Quotestd.rod standard report. This report is the Current Quote report in the Reports menu in views in the Quotes screen.

In Figure 16, a sample of the report displays quote master information at the top and provides a subreport in the middle containing the list of individual quote line items.



Current Quote SIEBEL

Date:	08/30/2001	**Sales Representative:**	SADMIN
Quote #:	1-10MLH	**Effective From:**	08/03/2001
Rev #:	1	**Valid Through:**	09/02/2001
Opportunity ID:		**Currency:**	USD

To:
Mike Dalton
Marriott International

Bill To:
Jamie Adams
Marriott International
10400 Fernwood Road
Bethesda, MD, 20817

Ship To:
Jamie Adams
Marriott International
10400 Fernwood Road
Bethesda, MD, 20817

Level	Item Name	Part #	Attributes	UOM	Net Price	Qty.	Ext. Price
1	Siebel Quotes	701-SEB-SSE-		Per User	$104.50	1	$104.50
2	Siebel Quotes	701-SEB-SSE-		Per User	$104.50	1	$104.50

Payment Method :
Name:
CC#: Exp Date:

Product Sub-Total:	$209.00
Services Sub-Total:	$0.00
Discounts:	$11.00
Tax:	$0.00
Freight Charges:	
Total:	$209.00

Figure 16. Current Quote Report Sample Page

Adding Global Variables to the Report Design

To make the field values in the master data row available to the subreports, you need to define a global variable that holds the contents of the master data row. This is defined as a public static variable on the report design component, and its type is defined as having the same class as the master data row.

To create a global data row variable in the report design

1. Expand the master datastream to view its child data row and note the data row's name.

   In the case of the Current Quote report (QUOTESTD), this name is ssQuoteDataRow, and the class is the same as the name because it has not been subclassed.

2. Navigate to the report design, which is the top-level component, and right-click it. In the pop-up menu, select Properties.

3. In the Properties window for the report design component, click the Variables tab.

4 Click New. The Class Variable dialog box appears.

5 Enter the following values:

- In the Name field, enter MasterRow.
- In the Type field, enter the previously noted data row class name.
- Leave Externally Defined Data Type unchecked.
- In the Storage radio button group, click Static.
- In the Visibility picklist, select Public.

6 Click OK to save the new variable.

Modifying the Master Datastream Component

In this step, the master datastream component is modified to obtain the data for its data row component from the global data row variable. This is accomplished by overriding the Fetch method on the master datastream.

To override the Fetch method on the master datastream

1 Navigate to the master datastream component (the datastream for the master report, rather than the subreport) and double-click it.

In the Quotestd.rod report, this component is ssQuoteQueryAddGlobalRow.

The Component Editor appears.

2 In the Component Editor, choose the Methods tab.

3 Select the Fetch method and click Override.

4 Replace the existing code in the Method Editor with the following code (using the relevant master data row name, if other than ssQuoteDataRow):

```
Function Fetch( ) As AcDataRow

Dim aMasterRow As ssQuoteDataRow


   Set aMasterRow = Super::Fetch( )

   If aMasterRow Is Nothing Then

      Exit Function

   End If


   Set MasterRow = aMasterRow
```

```
        Set Fetch = MasterRow

    End Function
```

5. Click Close.

The code in the Fetch method sets the MasterRow global variable to the contents of the row returned by the Super::Fetch operation, after first determining that the fetched record is not empty. Each time a master datastream record is obtained, the record in memory in MasterRow is replaced.

Referencing Global Variables in Controls

Let’s say, for example, that the name of the account needs to be referenced in each line item row.

To add the account name to line item rows

1. Navigate to the content frame QuoteItemContent1 in the subreport.
2. Make space for an account name between two fields.
3. Open the sscustom.rol library, if it is not already open, and drag the ssTxtS component from the Library Browser window to the desired location in the content frame.

   The Component Properties dialog box opens for the new text control.
4. In the ValueExp property in the Component Properties dialog box, enter the following value:

   ```
   QUOTERPT::MasterRow.ssAccount
   ```

   This sets the text control to display the current value of the ssAccount field in the MasterRow global variable, which stores the current quote.

To use a value from the global variable in a text control, use the ValueExp property of the control, as in the example. The expression is of the following form:

GlobalVar.FieldName

For example:

```
QUOTERPT::MasterRow.ssAccount
```

Debugging Tips for Composite Datastreams

The report design’s MasterRow variable must be Static with Public visibility.

The overridden Fetch method must call Super::Fetch() before using it to create the global data row. If unexpected results occur, assign a breakpoint to the Fetch method (select the line to breakpoint and press F9) and inspect all report variables (Debug > Variables). Make sure that the data row variable aMasterRow is being populated as expected.

The BuildFromRow method on a control can also be overridden for the purpose of using breakpoints to check the value of the MasterRow variable (and its fields) in order to confirm that MasterRow is exposing itself to the control.

10 Sorting Records in Memory

Generally, it is possible to obtain the kind of sorting and grouping that is necessary for a report using sort specifications and group breaks, as explained in Chapter 7, "Reports with Group Sections." However, when master data is sorted by the contents of a multi-value field (all records, not only the primary), or the master is in a many-to-many relationship with the detail, sorting cannot be defined in a sort specification. In these situations, the data must be passed to Actuate as a master-detail set of datastreams, and additional processing must occur in Actuate through the mechanism of a sort data filter.

This chapter explains how to design reports that require merging of records rather than a simple sort, and consists of the following topics:

About Report Sorted on a Multi-Value Field

Memory Sorting is employed when master records must be sorted by the contents of a multi-value field. An example of the situation is the Opportunities by Sales Rep report, which is obtained in the Opportunities screen by clicking the Reports button and selecting By Sales Rep from the drop-down list.

Opportunity records are listed in this report, sorted by the sales representative responsible for working on each opportunity. If there were only one sales representative per opportunity, this requirement could be satisfied with a sort specification and a group break. However, an opportunity record is assigned to a sales team, rather than a single sales representative. For you to see all the opportunities for each sales representative, the same opportunity must be listed under the name of each sales representative who is on that opportunity's sales team.

The relationship between an opportunity and the sales representatives assigned to it is a one-to-many relationship. In the Siebel business model, this is defined as a master-detail relationship (link) between the Opportunity and Position business components in the context of the Opportunity business object. The multi-value group displayed in Siebel Sales for opportunity records in the Sales Rep field is defined through the Position multi-value link.

For a report of this kind, the datastream must provide both the master and detail records. If only the opportunity records were sent to Actuate, it would be impossible to determine which sales reps were assigned to each opportunity; only the primary sales representative would be available for each opportunity. To send an interrelated set of opportunities and their detail sales representatives, the datastream must be defined in Siebel Tools with a report object definition for the Opportunity business component and a child subreport object definition for the Position business component. You can verify in Siebel Tools that this has been done for the Opportunities - By Sales Rep report.

Based on the master and detail records in the two datastreams, a set of merged records is created from the two business components that represents their cross-product, consisting of one merged opportunity record for each combination of an opportunity and a position in that opportunity's Sales Rep multi-value field. It is the merged records, created in memory, that are sorted into sales rep order and printed with sales rep sort breaks.

This report is analyzed in greater detail in "Examining a Report Sorted on a Multi-Value Field (MVF)" on page 124.

About Using Memory Sorting in a Many-to-Many Relationship

Memory sorting is also employed when master and detail business components have a many-to-many relationship, such as between opportunities and contacts or between accounts and opportunities. An example of this situation is the Contacts By Opportunity report, which is obtained in the Contacts screen by clicking the Reports button and selecting By Opportunity from the drop-down list.

The situation in this report is very similar to the one in the Opportunities by Sales Rep report, in that there are multiple detail records for each master, and the master records are to be sorted by a field in the detail records. You want to print contacts sorted by opportunity, which requires looking at the many-to-many relationship from the perspective of one opportunity to many contacts.

The set of object definitions in Siebel Tools uses the Opportunity business component as the master and includes a contact subreport. In the report design, merged opportunity records are created in a memory structure with the contact name included as an extra field. They are then sorted into contact order and printed with contact group breaks.

Notice that this could be switched around to create a report that prints contacts sorted by opportunity. The exported data supply library would use Contact as the business component in the master report object definition, and Opportunity in the subreport. In the report design, the structure in memory would hold the contents of each contact record plus an opportunity name field. The contact records would sort in opportunity order, with opportunity group breaks.

This report is analyzed in greater detail in "Examining a Report Based on a Many-to-Many Relationship" on page 131.

How a Memory Sort Report Works

This section provides background information that is necessary for understanding memory sort reports, then analyzes two existing standard reports that use this methodology.

About Data Filters

A datastream consists of two types of data adapters: datasources and data filters. Datasources retrieve data from an input source—the COM interface in the case of Siebel reports—and create data rows from the incoming records. Data filters sort, filter, or perform other computations on data rows. While a datastream must have at least one datasource, it is not required to have any data filters, and in the majority of standard reports data filters are not used.

A data filter receives data rows from one or more datasources or other data filters. A data filter processes the data it receives, then passes it to the next data filter (if there is one) or to the report.

Actuate provides three classes of data filters, one for accepting data from one datasource or data filter, one for merging data from multiple datasources and filters, and one for sorting. In the sscustom library, the classes for these three purposes are ssSingleInputFilter, ssMultipleInputFilter, and ssMemoryDataSorter, respectively.

Sort-merge requirements go beyond the capabilities of the standard sorting and grouping methodology and are handled with a data filter subclassed from ssMemoryDataSorter. This class inherits from the baseMemoryDataSorter class in the sssiebel library and ultimately from AcMemoryDataSorter in the Actuate Foundation classes. A data filter based on ssMemoryDataSort is also called a sort filter.

About Memory Structures

To sort data by the child business component, you must extract data from the master business component, store it, and re-sort it according to the values in the detail business component. The master and detail business component records are obtained through the master and detail datastreams.

In the Opportunities by Sales Rep (OPSLREP) standard report, each row of the opportunity query is processed before the data is formatted. Because data rows are transient objects, they are written into a temporary structure in memory as they are processed.

A global list or memory data buffer component can be used for this purpose. In the Opportunities by Sales Rep report, data rows from the Opportunities query are stored in a global list component. The AcList class, from which global list components are derived, is a smart array structure. A memory buffer (based on the AcMemoryBuffer class) is similar to a global list, but exhibits more complex behavior that is not required for most Siebel reports.

The RowList variable is derived from AcList, the AFC class designed to hold an ordered collection of objects. AcList is set up to function globally and has several methods that make it easy to manage lists. Methods for AcList variables are listed and described in Table 9.

Table 9. Methods Available for AcList Variables

Method	Class Derived From	Description
AddToHead	AcList	Adds an item to the beginning of the list.
AddToTail	AcOrderedCollection	Adds an item to the end of the collection.

Table 9. Methods Available for AcList Variables

Method	Class Derived From	Description
Contains	AcList	Returns TRUE if the list contains the item.
Copy	AcList	Copies the contents from another list to the end of this list.
GetAt	AcOrderedCollection	Returns the item at the specified location in the collection.
GetCount	AcCollection	Returns the number of objects in the collection.
GetHead	AcOrderedCollection	Returns the first item in the collection.
GetIndex	AcList	Returns the position of the node specified in the list.
GetTail	AcOrderedCollection	Returns the last item in the collection.
InsertAfter	AcList	Inserts a node in the list after the specified node.
InsertBefore	AcList	Inserts a node in the list before the specified node.
IsEmpty	AcCollection	Reports whether the collection is empty or not.
NewIterator	AcCollection	Creates an iterator for the collection.
Remove	AcCollection	Removes a specified item from the collection.
RemoveAll	AcCollection	Removes all contents from the collection.
RemoveHead	AcList	Removes the first node from the list.
RemoveTail	AcOrderedCollection	Removes the last item in the collection.

See the *Actuate Foundation Class Reference* for additional information on these methods.

About the Structure of the Report Design

Structurally, a memory sort report is similar to a report with group breaks, as described in Chapter 7, “Reports with Group Sections,” with some additional components. These include:

- A second report section
- A sequential section to group the two report sections
- A data filter in the new report section to reprocess data from the master and detail datastreams in the preceding report section

Figure 17 illustrates the structure of this kind of report.

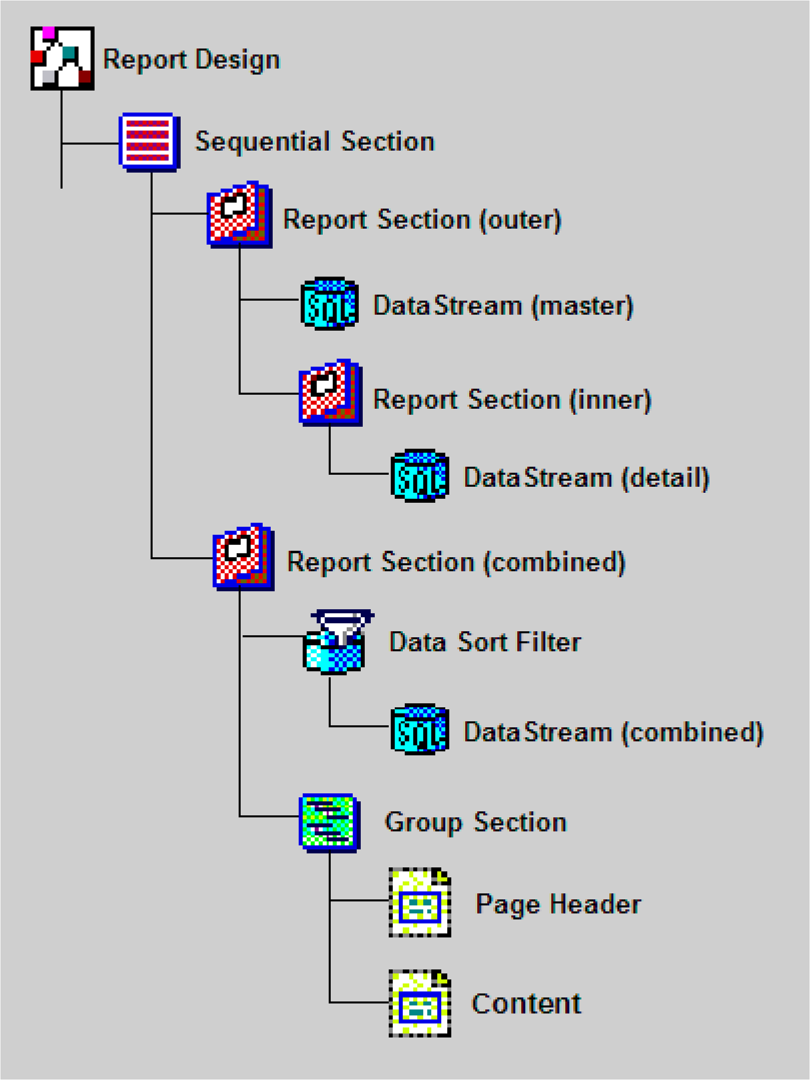


Figure 17. Memory Sort Report Structure

The major components in this structure are as follows:

- **Sequential Section.** This section groups together the outer report section, whose purpose is to obtain the master and detail records without generating any report lines, and the combined report section, which sorts and processes those records into a printed report. The sequential section causes these two processes to take place one after the other, which is necessary because the first process generates the stored set of records that is processed in the second process.
- **Outer Report Section.** This section holds the datastream for the master report.

- **Master Datastream.** This datastream obtains data records from the master business component. It is obtained from the data supply library file for the report.
- **Inner Report Section.** This section holds the datastream for the subreport.
- **Detail Datastream.** This datastream obtains data records from the detail business component for each master record. It is also obtained from the data supply library file.
- **Combined Report Section.** This section holds the data filter that obtains the merged records from the datastreams in the preceding report section and processes them into the appropriate sort order for the report. This report section also holds the content nodes that define the report output.
- **Data Sort Filter.** The data filter processes merged records from the datastreams in the preceding report section into the correct sorted order for the report output.
- **Combined Datastream.** This datastream is a child component of the data sort filter. It obtains merged records from the global list. The merged records have a record structure defined in the combined datastream's child data row component.
- **Group Section.** This section implements sort breaks between groups of like records.

Examining a Report Sorted on a Multi-Value Field (MVF)

A good MVF-sorted report to study is the Opportunities by Sales Rep report, which is invoked in the Opportunities screen (typically from the My Opportunities filter). You should examine the report output in Siebel Sales, and the report design in Actuate e.Report Designer Professional, leaving both open to compare them.

To generate the Opportunities by Sales Rep report

1. Open Siebel Sales.
2. Click Site Map and navigate to Opportunities > My Opportunities.
3. Click the Reports button, and from the drop-down list select By Sales Rep.

To open the report design for the Opportunities by Sales Rep report

1. Open Actuate e.Report Designer Professional.
2. Choose File > Open.
3. In the Open dialog box, navigate to the C:\Siebdev\RPTSRC\ENU\STANDARD folder (or the equivalent on your computer) and select Opslsrep.rod.

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

The Opslsrep.rod report design file in Actuate e.Report Designer Professional is shown in Figure 18.

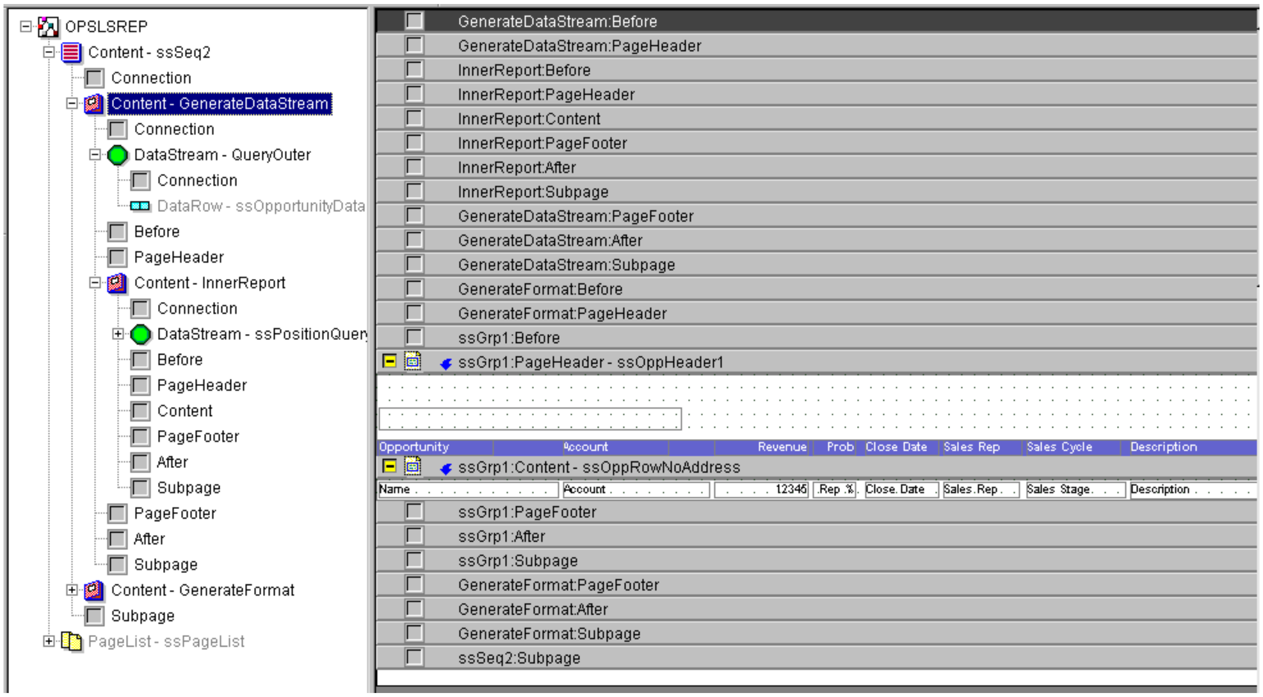


Figure 18. Opportunities by Sales Rep Design

Explore the report design, compare design elements to the corresponding features in the report output, and right-click components to view their property lists.

Notice the following features of this report design:

- There are three datastreams, corresponding to the master report, the subreport, and the combined report. The master report (GenerateDataStream) and the subreport (InnerReport) are strictly for data gathering and do not have content frames.
- If you open Siebel Tools and navigate to the report object definitions for the Opslsrep report (Opportunities - By Sales Rep) and its children, you can see that opportunity records are the parent, and position records (with a limited number of fields exposed) are the child. This is also reflected in the report design in Actuate, with the master datastream obtaining opportunities and the detail datastream obtaining positions for each opportunity.
- If you right-click the top-level report design component, choose Properties, and click the Variables tab, you can see that a custom variable has been defined. It is called RowList and is of the AcList class. This list structure variable holds the data rows as they are obtained from the input datastreams, for later access by the sort (combined) datastream. This is described in "Creating a Global List Variable" on page 126.
- The Fetch method in each of the three datastreams has been locally overridden in order to change the record acquisition code. You can determine this by opening the Properties window for the datastream, clicking the Methods tab, and noting the dark font used for the name of the Fetch method. Double-click the method name to view the code.

  - The Fetch method on the master datastream (QueryOuter) moves the data in the current master record to a public data row variable called OutSideDataRow. This variable is defined on the master datastream, making the current master record available to the detail datastream, ssPositionQuery. See "About the Fetch Method on the Master Datastream" on page 127.
  - The Fetch method on the detail datastream (ssPositionQuery) joins the master and detail data rows to create a combined data row, which is subsequently added to the RowList global list. See "About the Fetch Method on the Detail Datastream" on page 127.
  - The Fetch method on the combined datastream (GetFromList) gets each data row from the global list and passes the data row to the sort filter. See "About the Fetch Method on the Combined Datastream" on page 129.

- The combined report section, GenerateFormat, uses the GetFromList datastream that has been locally created and defined, rather than imported from the data supply library file generated by Siebel Tools. The parent component of this datastream, rather than the report section itself, is a data filter component. The datastream is created from the ssDataSource library component in sscustom.rol and the data filter from the ssMemoryDataSorter in the same library.

  The Compare method on the sort data filter re-sorts the data rows that have been sent to the filter. The sort data filter component is described in "About the Compare Method on the Sort Data Filter" on page 130.

Creating a Global List Variable

A global list variable, based on the AcList class in the AFC library, holds the data rows as they are obtained from the input datastreams, for later access by the sort datastream. In the Opportunities by Sales Rep report, this variable is called RowList. You add the global list variable to the top-level component in the report design, namely the report design itself.

To create a global list variable in the report design

1. Navigate to the Report Design, which is the top-level component, and right-click it. In the pop-up menu, select Properties.
2. In the Properties window for the Report Design component, click the Variables tab.
3. Click the New button. The Class Variable dialog box appears.
4. Enter the following values:
   - In the Name field, enter RowList.
   - In the Class field, select AcList.
   - Leave Externally Defined Data Type unchecked.
   - In the Storage radio button group, click Static.
   - In the Visibility picklist, select Public.
5. Click OK to save the new variable.

In the Opportunities By Sales Rep example, the RowList variable holds the merged data rows that were created in the GenerateDataStream report section. After each combined data row is built, it is added to the list by calling the AddToTail method on the RowList variable.

About the Fetch Method on the Master Datastream

The Fetch method on the master datastream (QueryOuter in the example report) moves the data in the current master record to a public data row variable, called OutSideDataRow. This variable is defined on the master datastream, making the current master record available to the detail datastream, ssPositionQuery. The Fetch method on the master datastream contains the following code:

```
Function Fetch( ) As AcDataRow


      Set Fetch = Super::Fetch()


' If a row is returned, then assign it to OutSideDataRow Var

   If NOT Fetch Is Nothing Then

      Set OutSideDataRow = Fetch

   End If


End Function
```

About the Fetch Method on the Detail Datastream

The Fetch method on the detail datastream joins the master and detail data rows to create a combined data row, which is subsequently added to the RowList global list. The field structure of the combined data row is defined in the data row child component of the combined datastream component. In the Opportunities By Sales Rep example, the combined data row is CombinedDataRow, and this is a child of the GetFromList datastream.

The Fetch method in the detail datastream has the following code:

```
Function Fetch( ) As AcDataRow

Dim aInsideDataRow As ssPositionDataRow

Dim aOutsideDataRow As ssOpportunityDataRow

Dim aCombinedDataRow As CombinedDataRow
```

```
' Initialize the List Object if it has been initialized

' This should only happen for the first time through

   If RowList Is Nothing Then

      Set RowList = New AcSingleList

   End If


' Get the current inside row

   Set aInSideDataRow = Super::Fetch( )

   If aInSideDataRow is Nothing Then

      Set Fetch = Nothing

      Exit Function

   End If


' Get a pointer to the Outside Data Row Variable, declared on the DataSource

   Set aOutsideDataRow = OPSLSREP::QueryOuter::OutSideDataRow

   If aOutsideDataRow Is Nothing Then

      Exit Function

   End If


' Create a new CombinedDataRow

   Set aCombinedDataRow = New CombinedDataRow


' Fill Combined Data row with entries from inner row

   aCombinedDataRow.ssSales_Rep = aInSideDataRow.ssLogin_Name

   aCombinedDataRow.ssPosName = aInSideDataRow.ssName

   aCombinedDataRow.ssClose_Date = aInSideDataRow.ssOpportunity_Close_Date

   aCombinedDataRow.ssClose_Date_Formatted=
aInSideDataRow.ssOpportunity_Close_Date_Formatted

   aCombinedDataRow.ssRevenue_Formatted =
aInSideDataRow.ssOpportunity_Revenue_Formatted

   aCombinedDataRow.ssRevenue= aInSideDataRow.ssOpportunity_Revenue
```

```
' Get values that are required from the outside data row
   aCombinedDataRow.ssName = aOutSideDataRow.ssName
   aCombinedDataRow.ssAccount = aOutSideDataRow.ssAccount
   aCombinedDataRow.ssCity = aOutSideDataRow.ssCity
   aCombinedDataRow.ssDescription = aOutSideDataRow.ssDescription
   aCombinedDataRow.ssPostal_Code = aOutSideDataRow.ssPostal_Code
   aCombinedDataRow.ssRep__ = aInSideDataRow.ssOpportunity_Rep__
   aCombinedDataRow.ssSales_Stage = aOutSideDataRow.ssSales_Stage
   aCombinedDataRow.ssState = aOutSideDataRow.ssState
   aCombinedDataRow.ssStreet_Address = aOutSideDataRow.ssStreet_Address

' Handle inside row so that Fetch continues to function
   Set Fetch = aInsideDataRow

' Add the newly created combined datarow to the Global list of rows
   RowList.AddToTail(aCombinedDataRow)

End Function
```

About the Fetch Method on the Combined Datastream

The Fetch method on the combined datastream gets data from the global list. For each row in the global list, the GetAt method on the global list variable is invoked to obtain the data, and the AddRow method on the parent data sort filter component is invoked to pass the data row to the filter.

The code for the Fetch method on the combined datastream is as follows:

```
Function Fetch( ) As AcDataRow
Dim curList As AcList
Dim curDataRow As CombinedDataRow

' Acquire a reference variable to the global RowList from first report
```

```
      Set curList = RowList
      If curList is Nothing then
         'MsgBox "failure to acquire Row List"
          Exit Function
      End If

   ' Set the data row to the Item in the list at the current position
      Set curDataRow = curList.GetAt(Position)
      If curDataRow Is Nothing Then
         Exit Function
      End If

      Set Fetch = curDataRow
      AddRow (Fetch)

   End Function
```

About the Compare Method on the Sort Data Filter

The Compare method on the sort data filter re-sorts the data rows that have been sent to the filter. A sort filter is configured primarily by overriding the filter's Compare method to specify how the rows should be sorted. Compare takes two rows as arguments and returns one of the following values:

- A positive number if the first row goes after the second row.
- Zero if both rows are the same.
- A negative number if the first row goes before the second row.

The code you define in the Compare method defines the criterion for ordering the records.

The Compare method calls the CompareKeys method (in the AcDataSorter class in the AFC) to compare two key values obtained from the same column. The CompareKeys method returns one of the following values:

- –1 if key1 is less than key2
- 0 if key1 equals key2
- 1 if key1 is greater than key2

In the case of the Opportunities By Sales Rep report, opportunities are being ordered primarily by sales rep and secondarily by revenue. That is, within each group with the same sales rep, the opportunities are ordered by revenue. This makes it possible for the group section to provide a group break at each change in sales rep, and within each group for the opportunities to be listed from highest to lowest in revenue.

The code for the Compare method on the sort data filter is as follows:

```
Function Compare( row1 As AcDataRow, row2 As AcDataRow ) As Integer

   Dim aRow as CombinedDataRow

   Dim bRow as CombinedDataRow


   Set aRow = row1

   Set bRow = row2


' Order By Sales Reps

   Compare = CompareKeys(aRow.ssSales_Rep, bRow.ssSales_Rep)


' Order by Revenue, descending

' May want to use the functional data here

   If Compare = 0 Then

      Compare = CompareKeys(Val(bRow.ssRevenue), Val(aRow.ssRevenue))

   End If


End Function
```

Examining a Report Based on a Many-to-Many Relationship

A good many-to-many report to study is the Contacts By Opportunity report, which is invoked from a view in the Contacts screen (typically the My Contacts view). You should examine the report output in Siebel Sales and the report design in Actuate e.Report Designer Professional, leaving both open to compare them.

To generate the Contacts By Opportunity report

1 Open Siebel Sales.

2 From the Site Map button, navigate to Contacts > Contacts List.

   My Contacts is the default view.

3 Click the Reports button, and from the drop-down list select By Opportunity.

   The Contacts By Opportunity report appears in a new web browser window.

To open the report design for the Contacts By Opportunity report

1 Open Actuate e.Report Designer Professional.

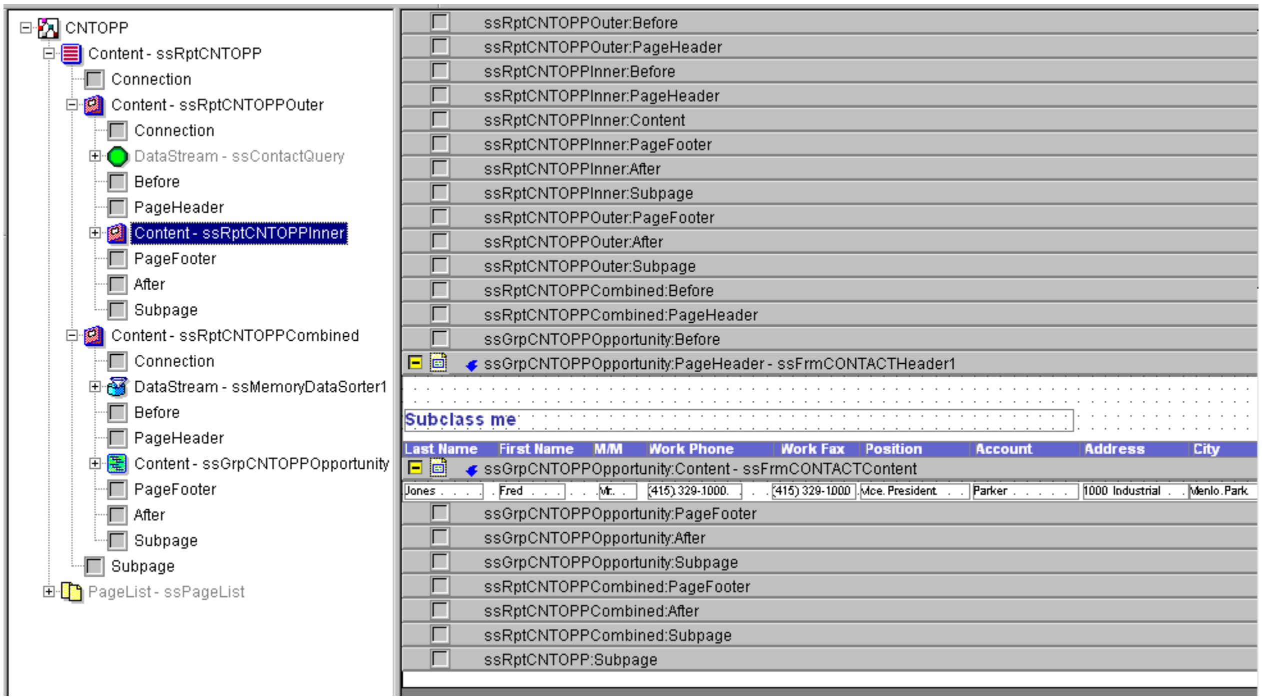
2 Choose File > Open.

3 In the Open dialog box, navigate to the C:\Siebdev\RPTSRC\ENU\STANDARD folder (or the equivalent on your computer) and choose Cntopp.rod.

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

The Cntopp.rod report design file in Actuate e.Report Designer Professional is shown in Figure 19.



```
CNTOPP
  Content - ssRptCNTOPP
    Connection
    Content - ssRptCNTOPPOuter
      Connection
      DataStream - ssContactQuery
      Before
      PageHeader
      Content - ssRptCNTOPPInner
      PageFooter
      After
      Subpage
    Content - ssRptCNTOPPCombined
      Connection
      DataStream - ssMemoryDataSorter1
      Before
      PageHeader
      Content - ssGrpCNTOPPOpportunity
      PageFooter
      After
      Subpage
    Subpage
  PageList - ssPageList
```

```
ssRptCNTOPPOuter:Before
ssRptCNTOPPOuter:PageHeader
ssRptCNTOPPInner:Before
ssRptCNTOPPInner:PageHeader
ssRptCNTOPPInner:Content
ssRptCNTOPPInner:PageFooter
ssRptCNTOPPInner:After
ssRptCNTOPPInner:Subpage
ssRptCNTOPPOuter:PageFooter
ssRptCNTOPPOuter:After
ssRptCNTOPPOuter:Subpage
ssRptCNTOPPCombined:Before
ssRptCNTOPPCombined:PageHeader
ssGrpCNTOPPOpportunity:Before
ssGrpCNTOPPOpportunity:PageHeader - ssFrmCONTACTHeader1
Subclass me
Last Name  First Name  M/M  Work Phone  Work Fax  Position  Account  Address  City
ssGrpCNTOPPOpportunity:Content - ssFrmCONTACTContent
Jones  Fred  Mr.  (415) 329-1000.  (415) 329-1000  Vice. President  Parker  1000 Industrial  Menlo.Park
ssGrpCNTOPPOpportunity:PageFooter
ssGrpCNTOPPOpportunity:After
ssGrpCNTOPPOpportunity:Subpage
ssRptCNTOPPCombined:PageFooter
ssRptCNTOPPCombined:After
ssRptCNTOPPCombined:Subpage
ssRptCNTOPP:Subpage
```

Figure 19. Contacts By Opportunity Report Design

Notice the following features of this report design (especially in comparison to the Opportunities By Sales Rep report described in "Examining a Report Sorted on a Multi-Value Field (MVF)" on page 124):

- The structures of the reports and the code defined on the corresponding datastreams and data filters are very similar between the reports.
- The global list variable and the master row global variable are both defined on the report design component Cntopp.rod. In Oppslsrep.rod, the global data row variable was defined in datastream Fetch code, which accomplishes the same result. Defining the data row variable on the report design component is preferable for clarity in your design.
- The Compare method in Cntopp.rod sorts the merged records primarily on opportunity name and secondarily on contact last name. This is consistent with the sorting and grouping requirements of the final report.
- If you open Siebel Tools and navigate to the report object definitions for the Cntopp.rod report (Contacts - By Opportunity) and its children, you can see that contact records are the parent, and opportunity records (with only the Name field exposed) are the child. This is also reflected in the report design in Actuate, with the master datastream obtaining contacts and the detail datastream obtaining opportunities for each contact.

  This structure is analogous to what was done in the Opslsrep.rod object definitions and report design. The business component that is to be sorted and printed is the master. The business component that provides the sort key is the detail. While this may seem counterintuitive in the case of a many-to-many report, in which the detail business component provides the group headings and the master business component provides the detail report rows, it is consistent with the data model of the Contact business object. Since the report is invoked from views in the Contacts screen, where the Contact business object is active, it is not possible to obtain records through a master-detail relationship in which Opportunity is the master. To do so would provide no benefit in any case because the records of the two business components are merged before sorting.

11 Using Graphics in Reports

This chapter explains the techniques for incorporating graphics, such as bar charts and pie charts, in reports. The overall methodology is described, the correspondences between Siebel chart types and Actuate chart types are detailed, and the property settings for a chart are explained. For more information about creating charts, see the chapter on presenting data in charts in the *Developing Advanced e.Reports* and *Designing e.Reports* manuals in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

This chapter consists of the following topics:

About Using Graphics in Reports

A chart is a control that is positioned in a content frame. In a report that provides only a chart and no textual data, the chart control occupies the entirety of the main content frame in the report section. Another option is to place a chart control in the After frame in a report that has a group break, in which case the chart appears after each group of text rows.

Charts used in reports fall into two categories, depending on the relationship between bars or curve points and the underlying data, as follows:

- **Detail charts.** Each data point is plotted individually as a curve point or a bar. Both the x- and y-axes must be numeric. Detail charts have limited utility and are generally used only for scatter charts. Detail charts are based on the acDetailChart class in the Actuate Foundation Class (AFC) library.

  NOTE: The use of a component from the AFC library, rather than the sscustom library, is an exceptional situation and applies only to detail charts. AFC components should not be used directly in Siebel reports except in this specific situation.

- **Summary charts.** All data points in a given classification are combined, and an aggregate value is plotted for each curve point or bar. Summary charts support plotting data series, in which one set of bars or line chart points is provided for each legend label. They also support text label and date values on the x-axis. Summary charts are based on the ssSummaryGraph class in the sscustom library.

In either case, the component is dragged onto an empty frame, and the properties are set to configure the use of data. A detail chart control can be added from the Chart button. A summary chart must be added from the Library Browser window for the sscustom library.

NOTE: A third chart type, HLC (high-low-close), is also available, but is not described in this chapter. An HLC chart is added using the chart button, and you select the AcHLCChart class when prompted.

About Actuate-Supplied Chart Types

Actuate provides many of the same chart types as Siebel eBusiness Applications. This makes it possible to duplicate these kinds of graphics in Siebel views and the reports for those views. All the chart types listed in Table 10 are available for both detail and summary charts, but many will work correctly only in summary charts. Table 10 shows supported chart types in Siebel applications and Actuate and the correspondences between them.

Table 10. Siebel Chart Types and Corresponding Actuate Chart Types

Siebel Chart Type	Actuate Chart Type
2dBar	Chart2dBar (vertical)
2dHorizBar	Chart2dBar (horizontal)
2dLine	ChartLine
2dPie	Chart2dPie
2dScatter	ChartScatter
2dSpline	
2dStackedBar	Chart2dBar (vertical, stacked)
3dBar	Chart3dBar (vertical)
3dClusteredBar	Chart3dBar (vertical, clustered)
3dHorizBar	Chart3dBar (horizontal)
3dLine	
3dPie	Chart3dPie
3dSpline	ChartTape
3dStackedBar	Chart3dBar (vertical, stacked)
Combo	Chart2dArea
	Chart3dArea
	ChartHLC (High-Low-Close)
	ChartOHLC (Open-High-Low-Close)
	ChartCandleStick

About Chart Control Properties

The property settings for a chart control are set in the Component Editor. Access the Component Editor by right-clicking the chart, choosing Properties from the pop-up menu, and clicking the Properties tab. Summary and detail charts have the same set of available properties, with the exception of the properties in the Expressions category, which differ between the two types.

The properties for a chart control, arranged by property category (the folder you must expand to view the property), are described in Table 11.

NOTE: In Actuate 7, the colors that you would list in the General: ValuesColorList property in the Component Editor for charts must be comma separated. The correct format for this property is: RGB(90,173,173), RGB(120,120,188), RGB(177,64,187), RGB(63,141,89), RGB(213,0,0). The colors for your chart will not be produced if you do not separate the color listing using commas.

Table 11. Chart Properties by Property Category

Category	Property	Description
	TargetWindow Name	The name of the target window in which the contents of a hyperlink appear.
Advanced	LinkExp	The expression defining a hyperlink for this chart object.
	ObjectVariable	The name of an optional method in the frame that will point to this chart object.
Expressions (in AcDetailChart only)	XLabelExp	Contains a data row variable, in square brackets, from which a text x-axis label can be obtained for each point. Note that values are not aggregated for each label.
	XValueExp	Contains a data row variable, in square brackets, that supplies the X value for each data point. For example, in a scatter chart that plots revenue on the y-axis against sales stage on the x-axis, the setting for this property would be [ssSales_Stage].
	YValueExp	Contains a data row variable, in square brackets, that supplies the Y value for each data point. For example, in a scatter chart that plots revenue on the y-axis against sales stage on the x-axis, the setting for this property would be [ssRevenue].

Table 11. Chart Properties by Property Category

Category	Property	Description
Expressions (in ssSummaryGraph only)	CategoryExp	Contains a data row variable or expression that specifies the category, that is, the x-axis value that groups data. This should normally evaluate to an integer. For example, in a summary chart that groups opportunities by month, the category key would be: Year([ssClose_Date]) * 12 + Month([ssClose_Date]) –1
	CategoryLabelExp	Contains a data row variable or expression that provides the text label for each category. For example, in a summary chart that groups opportunities by month, the category label expression would be: [ssClose_Date_ Formatted]
	SeriesExp	Contains a data row variable or expression that specifies the series (z-axis) for the data point, that is, the value that determines which bar color or line chart curve the point appears in. This should normally evaluate to an integer.
	SeriesLabelExp	Contains a data row variable or expression that provides the text label to appear in the legend for each series.
	YValueExp	Contains a data row variable or expression that supplies the Y value for each data point. For example, in a summary chart that displays opportunity revenue by month, the value expression would be: Sum(Val([ssRevenue]))

Table 11. Chart Properties by Property Category

Category	Property	Description
General	BackgroundColor	Background color of the chart.
	ChartBackgroundColor	Color of the rectangle around the chart.
	ChartBorderStyle	Style for the border around the chart. Options are ChartNoBorder, ChartSolidBorder, ChartDropShadow, ChartShadowAndBorder, ChartRaisedBorder, and ChartLoweredBorder.
	PointLabelAlignment	Enumerated. The options are: ■ ChartHorizontalAlignment aligns the point labels horizontally. ■ ChartVerticalAlignment aligns the point labels vertically. ■ ChartAutoAlignment tilts the data point labels such that they do no overlap.
	PointLabelColor	The color of the labels displayed for each point.
	PointLabelStyle	The style of the labels displayed for each point. Options are ChartNoPointLabels, ChartCustomLabels, ChartNumericLabels, ChartColoredNumericLabels, and ChartColoredCustomLabels.
	ValuesColorList	A comma-separated list of color names or RGB values to identify different data series.
ChartType-Specific	BarGrouping	Specifies that a bar chart is clustered, stacked, or without clustering or stacking. Ignored for charts other than bar charts. Options are ChartBarNoGrouping, ChartBarCluster, ChartBarClusterZ (3D only), ChartBarStack, and ChartBarStackPercentage.
	BarOrientation	Specifies that a bar chart has horizontal or vertical bars. Ignored for charts other than bar charts. Options are ChartBarVertical, ChartBarHorizontal, and ChartBarHorizontalReversed.
	ChartType	Specifies the chart type. Option are Chart2DPie, Chart3DPie, Chart2DBar, Chart3DBar, ChartLine, Chart2DArea, Chart2DScatter, ChartHLC, ChartTape, ChartOHLC, Chart2DArea, and Chart3DArea.

Table 11. Chart Properties by Property Category

Category	Property	Description
	HLCBarStyle	HLC and OHLC charts only. Determines how the chart displays high, low, and close tick marks. Options are ChartHLCAllBars, ChartHLCThickBars, ChartHLCNoClose, and ChartHLCNoBars.
	LineStyle	Specifies that a line chart has solid lines of default thickness, broken lines of a particular pattern, or thick lines. Applies to line charts only. Options are ChartDefaultLines, ChartPatternLines, and ChartThickLines.
	LineStylesList	Line charts only. A comma-separated list of line styles, one style for each data set. Options are SingleLine, DashLine, DotLine, DashDotLine, DashDotDotLine, and NullLine.
	LineThickness	Line and scatter charts only. Specifies the thickness of lines in chart-specific units.
	PointsArePercent	Area charts only. TRUE indicates that the numbers on the data points are percentage figures. FALSE indicates that the points are absolute numbers.
	ShowAsPercent	Area charts and pie charts only. Draws the chart showing the percentage of each data point as the sum of all points in that dataset. The area always fills the full chart height.
	ShowLines	Line, log/lin, lin/log, and log/log charts only. TRUE shows lines connecting the points.
	ShowSticks	Line, log/lin, lin/log, log/log, and scatter charts only. TRUE shows lines from the y-axis to each point.
	ShowSymbols	Line, log/lin, lin/log, log/log, and scatter charts only. TRUE shows symbols for each point.

Table 11. Chart Properties by Property Category

Category	Property	Description
HTML	Alignment	Specifies the alignment of the chart window within the flow of text in the report.
	AlternateText	In browsers that do not support Java applets, specifies the text that appears in place of the chart.
	BorderWidth	The number of pixels of thickness of the border around the chart.
	Margin	The number of pixels of horizontal and vertical space between the window and the surrounding text.
	UseDefaultSize	Determines whether to use the default image size computed by the browser. If it is set to TRUE, the system does not generate the height and width HTML attributes. If FALSE, the system generates these attributes from the Size property.
Legend	LegendBackgroundColor	Background color for the legend.
	LegendBorderStyle	The style of the box to draw around the legend. Options are ChartNoBorder, ChartSolidBorder, ChartDropShadow, ChartShadowAndBorder, ChartRaisedBorder, and ChartLoweredBorder.
	LegendColorText	TRUE if the legend entries are the same color as the lines, points, or bars they identify. FALSE if they are all the color given by Legend.LegendFont.Color.
	LegendFont	A group of properties that defines the font of the legend labels, including Bold, Color, FaceName, Italic, and so on.
	LegendPosition	The position of the legend. Defaults to ChartLegendRight.
Titles	TitleBackground Color	Background color for the rectangle enclosing the title.
	TitleBorderStyle	Style for the border around the title. Options are ChartNoBorder, ChartSolidBorder, ChartDropShadow, ChartShadowAndBorder, ChartRaisedBorder, and ChartLoweredBorder.
	TitleFont	A set of properties that defines the font for the chart title.
	TitleText	Title that appears above the chart.

Table 11. Chart Properties by Property Category

Category	Property	Description
TOC	TocAddComponent	Determines whether the component name is added to the report's table of contents. The values are: TOCAlwaysAdd. Always add the component to the table of contents. TOCIfAllVisible. Add component name to the table of contents only if the user can view at least one page generated from the component based on page security. This is the default. TOCIfAnyVisible. Add component to table of contents even if the user cannot view any pages generated from the component based on page security. TOCSkip. Never add the component to the table of contents. Use this to hide components such as parallel or sequential sections or detail frames from the user.
	TocAddContents	Determines whether the component’s contents are added to the report’s table of contents.
	TocValueExp	Returns a string to show as the table of contents entry for this object.
Viewing	CursorShape	The kind of cursor to show when the mouse cursor passes over the chart object.
	HelpText	The text to show for this chart object when the user asks for help.
	Searchable	Specifies whether the chart object can be searched in the Viewer.
	SearchAlias	The name to display to the user when building a search for this chart object.
	Selectable	TRUE if the user can select this chart object in the Viewer.
Visual	Position	The position of the chart object in its enclosing frame.
	Size	The size of the chart object.

Table 11. Chart Properties by Property Category

Category	Property	Description
X-Axis	XAxisPosition	The position of the x-axis. Options are ChartXAxisAuto, ChartXAxisTop, and ChartXAxisBottom.
	XAxisColor	Color of the axis lines.
	XAxisOrigin	The location of the chart origin. Options are ChartZeroOrigin, ChartAutoOrigin, and ChartCustomOrigin.
	XLabelFont	A set of properties that defines the font for the x- axis values labels, and that of the data labels adjacent to each point.
	XLabelFormat	The format string used to create custom x-axis labels.
	XLabelsList	A comma-separated list of quoted x-axis values.
	XLabelStyle	The kind of labels to show along the x-axis. Options are the following: ChartNoLabels. Displays no labels. ChartAutoLabels. Displays labels computed automatically. ChartCustomLabels. Displays custom labels based on the information you enter in the XLabelsList property. ChartExpressionLabels. Displays labels computed from an expression you specify in the LabelExp property.
	XMajorGridStyle	The style of line to draw for the major grid lines. Options are SingleLine, DashLine, DotLine, DashDotLine, DashDotDotLine, and NullLine.

Table 11. Chart Properties by Property Category

Category	Property	Description
	XMajorTickCount	The number of ticks to display. For charts with X values, this is the number of ticks to display. For charts without X values, this is the frequency of the ticks (that is, points per tick).
	XMajorTickStyle	The kind of tick marks to include. Options are ChartNoTicks, ChartAutoTicks, and ChartCustomTicks.
	XMax	Sets the maximum x-axis limit. This value is computed when the chart is displayed. When the chart is being designed, this value is used for displaying sample data.
	XMin	Sets the minimum x-axis limit. This value is computed when the chart is displayed. When the chart is being designed, this value is used for displaying sample data.
	XMinorGridStyle	The style of line to draw for the minor grid lines.
	XShowMinorTicks	TRUE indicates that the chart displays minor tick marks. If so, there will always be five minor tick marks between two major tick marks.
	XTitle	The text of the label to show for the x-axis.
	XTitleBackGround Color	The background color of the x-axis title.
	XTitleBorderStyle	The style of the rectangle that encloses the x-axis title. Options are ChartNoBorder, ChartSolidBorder, ChartDropShadow, ChartShadowAndBorder, ChartRaisedBorder, and ChartLoweredBorder.
	XTitleFont	A set of properties that defines the font of the x-axis.
	XValueSet	Indicates how the system should determine how many x-axis values are available. ChartDefaultXValues. No x-axis values are available; the Chart should provide default spacing along the x-axis. ChartXValuePerDataSet. There is one set of x-axis values provided in the points for the first (or only) data set that apply to all points. ChartXValuePerPoint. Each point has its own X value.
	XVerticalLabels	TRUE to show x-axis labels vertically instead of horizontally.

Table 11. Chart Properties by Property Category

Category	Property	Description
Y-Axis	DualYAxisColor	Color of the axis lines.
	DualYLabelFont	A set of properties that defines the font for the y- axis values labels, and that of the data labels adjacent to each point.
	DualYTitleFont	A set of properties that defines the font of the y-axis titles.
	YAxisColor	Color of the axis lines.
	YAxisPosition	The position of the y-axis. Options are ChartYAxisAuto, ChartYAxisLeft, and ChartYAxisRight.
	YAxisOrigin	The location of the chart origin. Options are ChartZeroOrigin, ChartAutoOrigin, and ChartCustomOrigin.
	YLabelFont	A set of properties that defines the font for the y- axis values labels, and that of the data labels adjacent to each point.
	YLabelFormat	The format string used to create custom y-axis labels.
	YLabelsList	A comma-separated list of quoted y-axis values.
	YLabelStyle	The kind of labels to show along the y-axis. Options are the following: ChartNoLabels. Displays no labels. ChartAutoLabels. Displays labels computed automatically. ChartCustomLabels. Displays custom labels based on the information you enter in the XLabelsList property. ChartExpressionLabels. Displays labels computed from an expression you specify in the LabelExp property.
	YMajorGridStyle	The style of line to draw for the major grid lines. Options are SingleLine, DashLine, DotLine, DashDotLine, DashDotDotLine, and NullLine.
	YMajorTickCount	The number of ticks to display. For charts with y-axis values, this is the number of ticks to display. For charts without y-axis values, this is the frequency of the ticks.
	YMajorTickStyle	The kind of tick marks to include. Options are ChartNoTicks, ChartAutoTicks, and ChartCustomTicks.

Table 11. Chart Properties by Property Category

Category	Property	Description
	YMax	Sets the maximum y-axis limit. This value is computed when the chart is displayed. When the chart is being designed, this value is used for displaying sample data.
	YMin	Sets the minimum y-axis limit. This value is computed when the chart is displayed. When the chart is being designed, this value is used for displaying sample data.
	YMinorGridStyle	The style of line to draw for the minor grid lines. Options are ChartDefaultLines, ChartPatternLines, and ChartThickLines.
	YShowMinorTicks	TRUE indicates that the chart displays minor tick marks. If so, there will always be five minor tick marks between two major tick marks.
	YTitle	The text of the label to show for the y-axis.
	YTitleBackGround Color	The background color of the y-axis title.
	YTitleBorderStyle	The style of the rectangle that encloses the y-axis title. Options are ChartNoBorder, ChartSolidBorder, ChartDropShadow, ChartShadowAndBorder, ChartRaisedBorder, and ChartLoweredBorder.
	YTitleFont	A set of properties that defines the font of the y-axis titles.
	YTitleOrientation	Determines how the text appears for the y-axis title. The default is ChartYLabelUp, which rotates the text 90 degrees counterclockwise. Options are ChartYLabelHoriz, ChartYLabelUp, and ChartYLabelDown.
	YVerticalLabels	TRUE to show x-axis labels rotated 90 degrees.

12 Smart Reports

This chapter describes the essential components of Smart Reports, which include specialized graphical elements and formatted summary information. In addition to the specialized graphical information and formatted summary, Smart Reports provide relevant detailed information. This chapter describes the implementation process, and provides examples to walk you through the process of creating them. This chapter is intended to allow you to configure and extend special graphical components, such as thermometers.

This chapter consists of the following topics:

About Smart Reports

Siebel Smart Reports are reports designed to serve the needs of senior managers. In this section their purpose is explained, the standard Smart Reports are listed and described, and the typical visual features are illustrated.

Purpose of Smart Reports

Typical reports generated from application software tend to be operational/transactional data spanning multiple pages. While this is generally appropriate for operations personnel, senior managers often need reports that go beyond the mere presentation of data and help extract the key information for decision making by bringing it into high visual relief.

In some organizations, high-level executives use reports as their primary form of interaction with Siebel eBusiness Applications, and reports provide their typical means of accessing an application's data. It is important to provide such users with reports that include summary information and performance metrics in graphical form in a concise and intuitive format. Siebel Smart Reports help senior managers users characterize a situation, analyze various issues, and take appropriate action.

For example, sales representatives, sales managers, and call center managers can use Smart Reports to measure their performance against recommended best practices, analyze trends, identify exceptions, and take appropriate action. Smart Reports are organized to start with summary information and proceed to the most detailed information. The contents fall into three sections or categories of information:

- **Dashboard section.** The first section in a Smart Report is a *dashboard*. The dashboard provides intuitive graphical indicators for the key measures of utility and an order of merit. This helps the reader gauge the situation being studied: whether an opportunity is worth pursuing, whether the pipeline is healthy, and so on.
- **Summary sections.** When the reader has identified the key issues, the reader can proceed to the *summary sections* and analyze them. These sections contain charts and diagrams that aid analysis. For example, a summary section may include a Pipeline versus Quota chart that helps pinpoint the region that is in danger of missing targets.
- **Detail sections.** Having analyzed the issue, the reader can drill into the underlying detail sections and take appropriate action. These tend to present information in a format similar to those of conventional standard reports. The manager can target key decision makers with activities that address their main decision issues, or identify the opportunities in the pipeline that need attention.

About Standard Smart Reports

Table 12 describes the standard Smart Reports provided with Siebel applications.

Table 12. Standard Siebel Smart Reports

Name	Description	Hierarchy of Data
Opportunity Detail	Describes the relevant details pertaining to the opportunity. It contains information about the relative importance of the opportunity, the likelihood of closing, the steps taken to date, and planned actions.	The dashboard summarizes Deal Size, Buying Influence, Probability of Closing, Competitive Activity, and Stage/Milestone. Summary and detail sections are provided for the following areas: Contacts, Products, Competitors, Decision Issues, Key Activities, All Activities, and Notes.
Account Summary	Describes the relevant details pertaining to the account. It contains information about the account's historical and future revenue, satisfaction, organizational hierarchy, and service requests.	The dashboard summarizes Past Revenue, Pipeline Revenue, Customer Satisfaction, and Competitive Activity. Summary and detail sections are provided for the following areas: Revenue Over Time (closed and pipeline), Current Opportunities, Contact Detail, Products Installed, Customer Satisfaction, Open Service Requests (by severity), and Campaigns/Activities.
Pipeline Analysis	An analysis of the current pipeline. It uses historical information to determine the revenue that is expected to be generated over the next four quarters and how it compares to the quota for that period.	The dashboard summarizes Expected Revenue Current Quarter and Expected Revenue Next Four Quarters. Summary and detail sections are provided for Pipeline Revenue and Opportunities Detail.

Table 12. Standard Siebel Smart Reports

Name	Description	Hierarchy of Data
Quota Summary	Measures performance relative to quota for the current period and year-to-date. It tracks closed as well as pipeline revenue to determine whether the goal for the quarter can be met.	The dashboard summarizes Revenue Current Quarter and Revenue YTD. Summary and detail sections include Revenue by Direct Report, Opportunities (closed), and Opportunities (open).
Service Request Aging Analysis	Analyzes the aging of the currently open service requests. It tracks three metrics that may explain performance, the number of open service requests, call volume, and average resolution time during the past months.	The dashboard summarizes Time Open (current SRs), Number of Open SRs, Call Volume, and Average Resolution Time. Summary and detail sections include Open Service Requests (by severity) and Closed Service Requests (by severity).
Account Service Detail	Summarizes the service-related information pertaining to the account. It contains information about currently open service requests, customer satisfaction, and the historical service request resolution for the account.	The dashboard summarizes Revenue, Open SRs, and Customer Satisfaction. The first summary section includes summary graphics for Severity Distribution, Aging by Severity, SR Substatus, and Customer Satisfaction. There is also a Closure Times by Severity summary section. There is one detail section listing Open Service Requests by severity.

About Visual Features for Smart Reports

Smart Reports are labeled as such in the Reports drop-down list in particular views. For example, from the Opportunities screen tab navigate to the My Opportunities view. Click Reports and review the drop-down list of reports.

Two important graphical elements specific to Smart Reports are described as follows:

- **Order of Merit indicator.** Reflected in Figure 20, an up, down, or right arrow that indicates the merit of the opportunity, account, and so on, based on one or more criteria being analyzed in the report. The direction of the arrow depends on whether the values measured by the thermometers are above or below their triggers. A right arrow indicates that two of the three thermometers are above their triggers. The up arrow represents that all three thermometers are above their triggers.

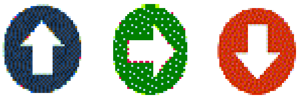


Figure 20. Order of Merit Indicators

- **Thermometer.** The vertical and horizontal partially filled rectangle graphs are called thermometers. Figure 21 shows the past revenue for an average account.

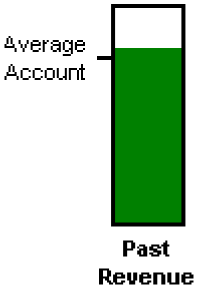


Figure 21. Thermometer

A thermometer is defined by four parameters:

- **Measure.** The quantity being measured by the thermometer. The value of this quantity determines the height of the shaded area within the rectangle.
- **Minimum.** The value represented by the bottom part of the rectangle.
- **Maximum.** The value represented by the top part of the rectangle.
- **Trigger.** A benchmark to which the value of the measure is compared. The order-of-merit indicator depends on whether the value of the measure is above or below the trigger.

For information on the configuration of thermometers, see "About Thermometers Used in Smart Reports" on page 156.

About the Report Structure for Smart Reports

Smart Reports have a fairly complex structure relative to other standard Siebel reports. The hierarchy of high-level report design components in a Smart Report is illustrated in Figure 22.

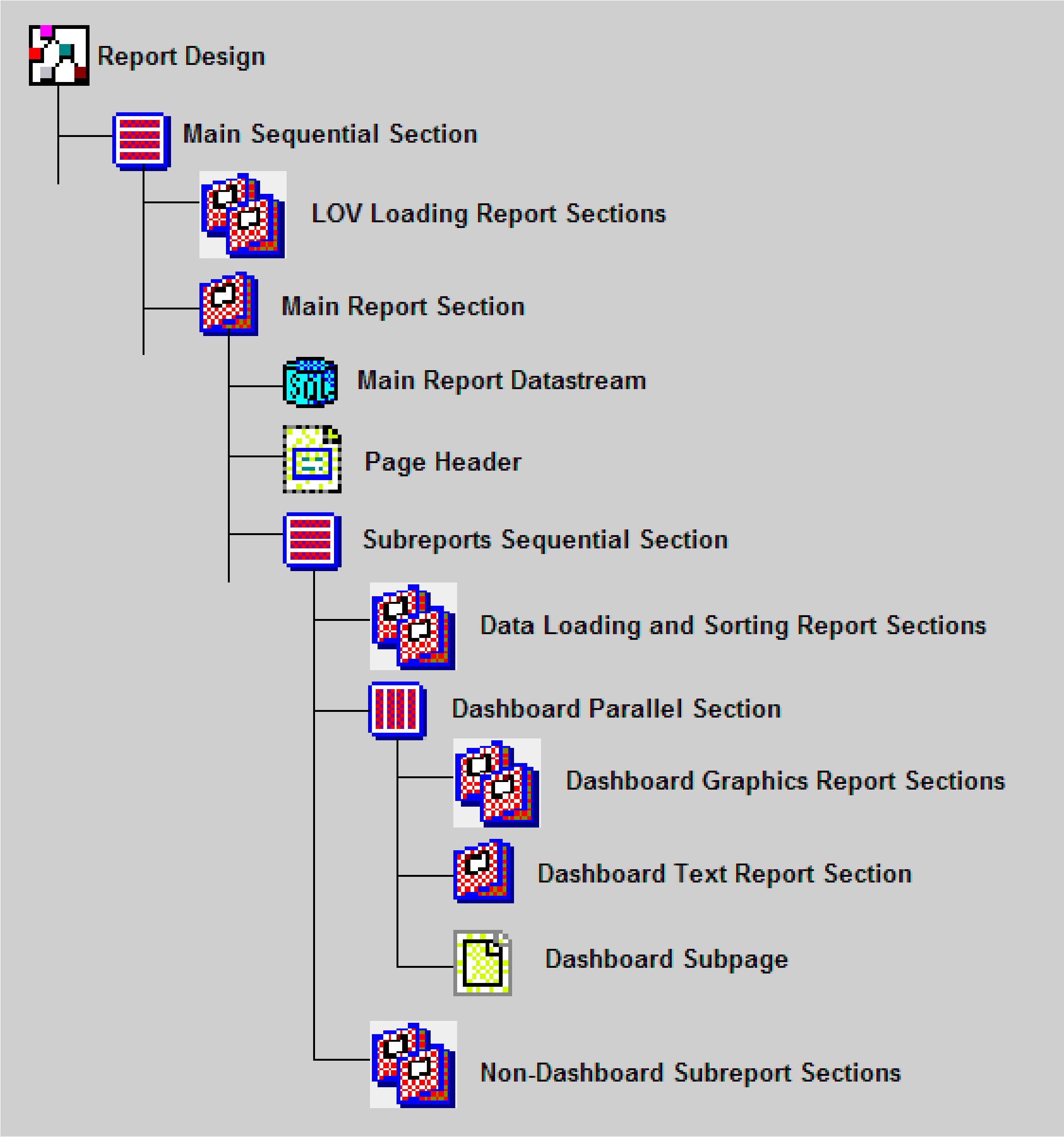

Report Design
Main Sequential Section
LOV Loading Report Sections
Main Report Section
Main Report Datastream
Page Header
Subreports Sequential Section
Data Loading and Sorting Report Sections
Dashboard Parallel Section
Dashboard Graphics Report Sections
Dashboard Text Report Section
Dashboard Subpage
Non-Dashboard Subreport Sections

Figure 22. Smart Report Component Hierarchy

The design components in Figure 22 on page 152 are as follows:

- **Main Sequential Section.** This is the parent component for all the major report sections. It causes its child sections to execute sequentially.
- **LOV Loading Report Sections.** This is the first set of report sections executed. The purpose of these report sections is to obtain high, low, and average (or trigger) values for each of the thermometers from records in the List of Values (LOV) table. For example, in the Account Summary Smart Report, there is one LOV loading section each for the Past Revenue, Pipeline, and Customer Satisfaction thermometers. Each of these report sections obtains a record from the List of Values table corresponding to a particular LOV type. For more information, see "Obtaining the Minimum, Maximum, and Trigger Values for Smart Reports" on page 157.
- **Main Report Section.** This report section holds all the design components for the Smart Report, other than the initial data collection sections and the pagelist. Its children are the main report datastream, the page header for the Smart Report, and the sequential section that processes the dashboard and various subreports.
- **Main Report Datastream.** This datastream is the master datastream for the report, providing opportunity records for the opportunity Smart Report, account records for the account Smart Report, and so on. It uses the data row from the master datastream in the data supply library file.
- **Page Header.** This component defines the page header for the Smart Report. In the page header, the name of each major entity (account, opportunity, and so on) appears, as it does in other standard reports. Certain summary values for the entity may be displayed as well. The unique feature of the page header in a Smart Report is that it also displays the order-of-merit indicator for that entity. The configuration of the order-of-merit indicator is discussed in "About the Order-of-Merit Indicator in Smart Reports" on page 155.
- **Subreports Sequential Section.** This section contains and sequentially processes the various report sections for the main report and subreports. These include data sorting sections, the dashboard section, and sections for the subreports that follow the dashboard in the report.
- **Data Loading and Sorting Report Sections.** These report sections are the first to appear in the sequential section, before sections that actually generate report lines and graphics. The data loading and sorting sections obtain and manipulate detail data for the current master record. The data is obtained from the various detail datastreams in the data supply library file.

  The detail datastreams accessed at this stage are those that must store data in global memory structures, often with some sorting or merging before storage. Detail datastreams without a global memory storage requirement (such as the contact, product, and activity datastreams in the Account Summary report) do not need to be included before the report sections in which they are used.
- **Dashboard Parallel Section.** A parallel section contains multiple report sections displayed in different flows on the same page, often side by side. To create a dashboard section displaying thermometers and other graphics and text side by side, a parallel section is required. The children of the parallel section are the report sections for graphics and text that appear in the dashboard.

- **Dashboard Graphics Report Sections.** These report sections correspond to individual graphic elements in the dashboard. Each has a datastream, and some have a data filter as well. Each also has a content frame. Included as a child of the content frame is either a graph component or a thermometer frame.

  A thermometer frame is derived from the ssThermometer component in the ssSmart.rol library. The OnRow method in a thermometer frame calculates values for, and passes four parameters to, the corresponding method in the parent library thermometer: MinimumValue, MaximumValue, TriggerDataValue, and DataValue. The thermometer is drawn based on these four values. The thermometer label, trigger label, and fill color are specified in properties in the thermometer frame. For information on thermometer configuration, see "About Thermometers Used in Smart Reports" on page 156.

  When a graph is displayed instead of a thermometer (such as the Competitive Activity graph in the Opportunity Detail and Account Summary reports), the content frame holds an AcDetailGraph or ssSummaryGraph control, configured as described in Chapter 11, "Using Graphics in Reports."

- **Dashboard Text Report Section.** This section includes the datastream and content frame for the textual master report information. The report text in the dashboard is from one master record; for example, in the Account Summary report, the fields for the account record are printed in the dashboard text report section. The datastream for this section is the master datastream in the data supply library file. The content frame provides blank space for the thermometers and other dashboard graphics, although these are positioned using the dashboard subpage rather than this content frame.

- **Dashboard Subpage.** A subpage establishes the physical layout of visual elements in a section. The dashboard subpage specifies the layout of the dashboard parallel section. The locations of the thermometers, dashboard graphs, and text report section are determined here by defining and positioning a flow for each in the subpage. Generally, the flow for the text report section occupies the entire subpage, and the other flows each occupy some portion of this area.

- **Nondashboard Subreport Sections.** The subreports that make up the summary and detail sections appear sequentially in the report following the dashboard parallel section. These are configured as typical subreports, each with a datastream and a content section. However, the data may be obtained from a memory structure (previously loaded in the data loading and sorting sections) and sorted or merged using a data filter.

About the Order-of-Merit Indicator in Smart Reports

The order-of-merit indicator appears at the top right of the first page in a Smart Report and is configured in the page header. It consists of an up, down, or right arrow that indicates the merit of the opportunity, account, and so on, based on one or more criteria being analyzed in the report. The direction of the arrow depends on whether the values measured by the thermometers are above or below their triggers. Figure 23 shows the three order-of-merit indicators. A right arrow indicates that two of the three thermometers are above their triggers. The up arrow represents that all three thermometers are above their triggers.

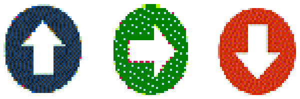


Figure 23. Up, Right, and Down Order-of-Merit Indicators

The order-of-merit indicator is derived from one of three predefined image components in the ssSmart.rol library: ssOrderOfMeritUpImage, ssOrderOfMeritDownImage, or ssOrderOfMeritPushImage. Code in the Finish method of the page header component determines, from the status of all thermometers relative to their targets, whether the arrow direction to display is up, down, or right.

For example, in the Account Summary report, the page header component is ssOrderOfMeritHeader1. The following code appears in the Finish method:

```
ArrowDirection = "DOWN"

If CustomerSatisfactionAboveTarget And PipelineThermometerAboveTarget Then

ArrowDirection = "UP"

End If

If PastRevenueThermometerAboveTarget And Not PipelineThermometerAboveTarget Then

ArrowDirection = "PUSH"

End If
```

In the case of this report, the order-of-merit indicator direction is based on the status of two thermometers, Past Revenue and Pipeline. If both are above target, the arrow direction is up. If Past Revenue is above target and Pipeline is not, the direction is right (called PUSH in the code example). If Past Revenue is below target, regardless of Pipeline, the direction is down.

The code that generates the Boolean values for the two xxAboveTarget variables is located in the OnRow methods of the respective thermometers, as described in "Obtaining the Data Value for Use with Smart Reports" on page 159.

Code in the Finish method of the page header determines the arrow direction to use and passes a value of UP, DOWN, or PUSH (the right arrow) in the ArrowDirection variable. Code in the Finish method of the page header's parent library component (ssOrderOfMeritHeader) loads and generates the arrow image for the correct direction, using image components in the ssSmart.rol library and bitmap files saved in C:\Siebdev\RPTSRC\ENU\LIB.

NOTE: If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

About Thermometers Used in Smart Reports

A thermometer is a partially filled rectangle graph in the dashboard section of a Smart Report. Figure 24 shows the parts of a typical thermometer. The parts are the trigger label, the mercury level, and the thermometer label.

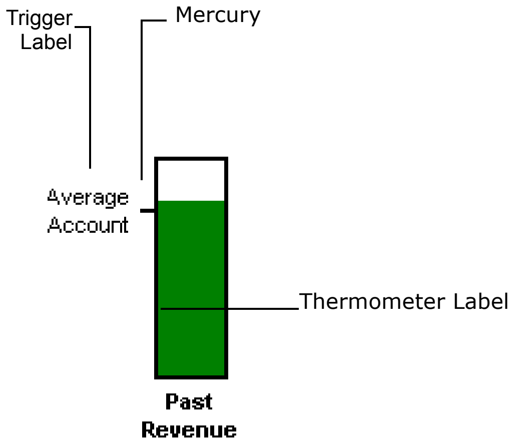


Figure 24. Parts of a Thermometer

A thermometer frame is derived from the ssThermometer component in the ssSmart.rol library by subclassing. Most of the functionality required for implementing a thermometer is already defined in the ssThermometer library component. A custom thermometer can be defined by adding an ssThermometer component to a frame in a report section; configuring three properties; and writing code to calculate four values and pass them, by way of global variables, to methods in the parent component.

The three properties to configure in the thermometer component are the following:

- **Color.** Color of the mercury in the thermometer. Each thermometer in a dashboard should use a different mercury color.
- **Thermometer Label.** The text of the label that appears beneath the thermometer.
- **TriggerLabel.** The text of the label that appears to the left of the trigger tick mark on the left side of the thermometer.

The four parameters that are passed to the ssThermometer library component are the following:

- **DataValue.** The value of the parameter being displayed. This determines the height of the mercury. For example, in a thermometer displaying past revenue for an account, the value of the past revenue of the current account would be displayed as a height relative to the minimum and maximum values for all accounts. Unlike the other three parameters, this one is calculated for the specific entity being reported on—in this case, one account rather than all accounts.
- **MinimumValue.** The minimum value for this entity among all like entities (accounts, opportunities, and so on). Determines the value associated with the bottom edge of the thermometer frame.
- **MaximumValue.** The maximum value for this entity among all like entities. Determines the value associated with the top edge of the thermometer frame.
- **TriggerDataValue.** The value that determines whether the data value for this entity is above, is below, or meets the target established for this kind of entity. In the Past Revenue example, the trigger data value establishes the height, relative to the maximum and minimum values of the trigger tick mark.

These four values are passed in the OnRow method in the thermometer component. They are passed, by way of global variables defined in the report design (top-level) object, to the corresponding method in the parent library thermometer, ssThermometer. The thermometer is drawn based on these four parameters and on the text and color properties specified in the thermometer component.

Much of the effort in configuring a thermometer lies in obtaining values for the minimum, maximum, and trigger from all records, and for the data value from the current record. The thermometer must also be positioned as a flow in the dashboard subpage. These configuration issues are explained in separate subsections.

Obtaining the Minimum, Maximum, and Trigger Values for Smart Reports

Various techniques are employed to determine the minimum, maximum, and trigger values for a thermometer. These values are constant through the entire report and all entities of the same type. For example, the minimum, maximum, and trigger values for the Past Revenue thermometer in the Account Summary report are constant for all accounts.

For many thermometers, the constant values are administered in List of Values Administration in a Siebel application. In the case of the Past Revenue thermometer, the three constants are set in the TARGET_ACCNT_LIFE_REV type. The minimum is set in the Target Low field, the maximum in the Target High field, and the trigger in the Order field. For information on configuring lists of values, see *Applications Administration Guide*.

Settings in the List of Values table in a Siebel application are communicated to Actuate reports through a datastream, similarly to the passing of business component records by way of the data supply library file. In the case of an LOV datastream, the appropriate datastream is subclassed from the ssList_Of_ValuesQuery datastream component in the ssSmart.rol library, rather than from a data supply library file.

The ssList_Of_ValuesQuery datastream component is subclassed for all LOV datastreams. The only differences between LOV datastreams lie in the name, the search specification—which is the means through which the desired LOV type is accessed—and the logic in the Fetch method.

The LOV report section for the Past Revenue thermometer provides a good example. The structure of this report section is illustrated in Figure 25.

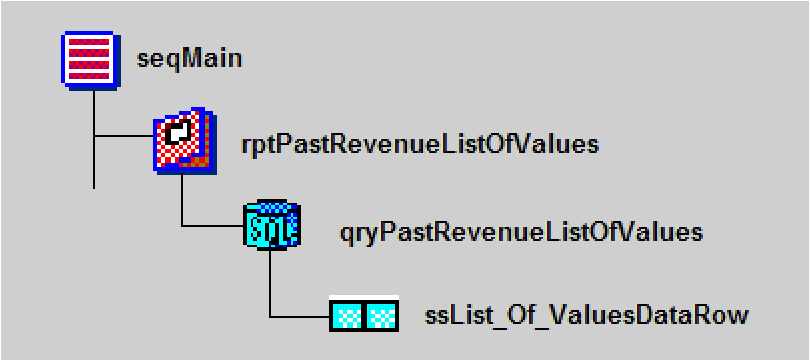


Figure 25. Structure of an LOV Data Loading Report Section

This report section consists of the following components:

- **Report section.** The LOV data loading report section—rptPastRevenuListOfValues in this case—serves no purpose other than to contain the LOV datastream. One such report section is provided for each LOV datastream to be defined. There will be one LOV datastream for each set of minimum, maximum, and target values to be obtained from an LOV record, generally one set of values per thermometer.
- **Datastream.** The LOV datastream—qryPastRevenuListOfValues in this case—is subclassed from the ssList_Of_ValuesQuery component in the ssSmart.rol library. The datastream is configured in the SearchSpec property and in the code in the Fetch method.
  - **SearchSpec property.** The text in this property specifies how to obtain the LOV record that provides the set of constants. This is based on the Type field in the LOV record. In the case of the Past Revenue thermometer, the search specification expression in the data loading datastream is the following:

    ```
    [Type] = 'TARGET_ACCNT_LIFE_REV'
    ```

- **Fetch method.** The code in the Fetch method specifies which fields provide the maximum, minimum, and target values in the LOV record and perform any necessary computations and conversions. The values obtained are passed to the global variables in the report design object that have been defined for the set of constants for one thermometer. For the Past Revenue thermometer, the Fetch method includes the following lines of code:

  ```
  PastRevenueAverage = Val( aRow.ssOrder_By )

  PastRevenueHigh = Val( aRow.ssTarget_High )

  PastRevenueLow = Val( aRow.ssTarget_Low )
  ```

- **Data row.** The data row component is always ssList_Of_ValuesDataRow. It is included in the datastream component when the datastream component is created in the report design by subclassing. The data row contains the set of input variables corresponding to the record structure of the LOV table in the Siebel application.

Obtaining the Data Value for Use with Smart Reports

The Past Revenue thermometer example (in the Acsum.rod [Account Summary] report) uses a total of past revenue for opportunities for the current account as its data value. The components and logic for deriving this data value are described here.

The components for deriving the data value occur in two areas in the report structure:

- An array of opportunity records is loaded in one of the data loading report sections for later access.
- A dashboard report section totals the relevant opportunity records and passes the total in the DataValue parameter to the thermometer component.

The components for loading the opportunity record array are in the rptCollectOpportunitiesAndThreats report section, near the beginning of secMain, as shown in Figure 26.

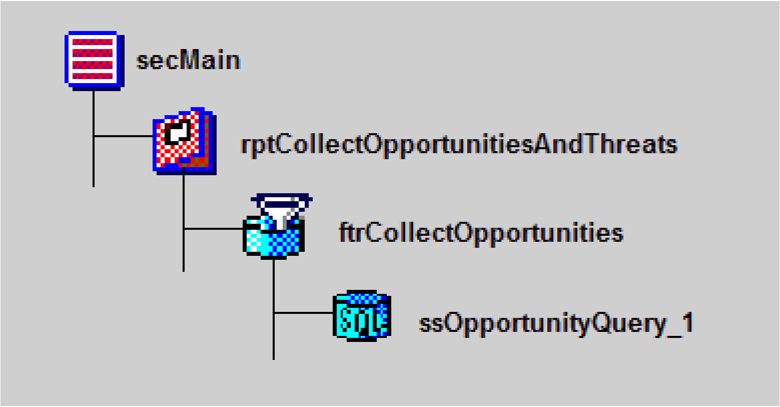


Figure 26. Components for Loading the Opportunity Array

This report section loads two global arrays (attached as variables to the top-level report design object), PastOpportunityArray and PipelineOpportunityArray. These both hold opportunity records for the current account, but meet different criteria. The former of these is used later in the report logic to derive the data value for the thermometer of interest (the latter is used for another thermometer, thermoPipeline).

The datastream that supplies this report section, ssOpportunityQuery_1, is a subreport datastream in the data supply library file for the report.

The processing logic for the report section is in the Start and Fetch methods for the filter component, ftrCollectOpportunities. The Start method empties the two arrays. The Fetch method sorts records from the opportunity datastream that match specified criteria into the two arrays. Records for PastOpportunityArray are those that have a Rep% (percentage) of 100; those for PipelineOpportunityArray have a Rep% lower than 100 and a close date later than the date the report is run.

The components and logic for determining the past revenue for the current account, based on the opportunity records stored in memory, are in the rptPastRevenueThermometer report section in the dashboard parallel section (parDashboard). This report section and its child components are illustrated in Figure 27.

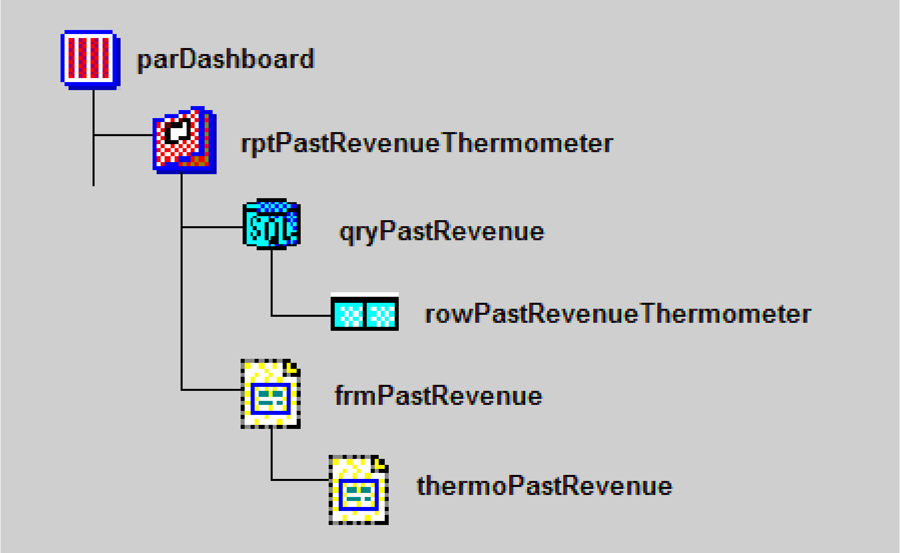


Figure 27. Thermometer Report Section Components

The components in Figure 27 on page 160 are as follows:

- **Report section.** The thermometer report section, rptPastRevenueThermometer, contains datastream and data row components for extracting a past revenue total for the account from opportunity records in memory. It also contains a container frame, frmPastRevenue, that holds the thermometer component, thermoPastRevenue. The thermometer component generates the thermometer based on the values passed to it in property settings and parameter variables.

- **Data row.** The data row component, rowPastRevenueThermometer, holds one variable: PastRevenue. A single row containing this value is all that the data row and datastream need to generate for the thermometer.
- **Datastream.** The Start and Fetch methods in the datastream component process the opportunity records in the PastOpportunityArray memory structure. An iterator, PastRevenueIterator, is set up for this array, and the ssFunctionalRevenue value in each opportunity record is accumulated in the PastRevenue variable. It is the value of this variable that the datastream provides to the thermometer as the DataValue parameter.
- **Content Frame.** The frmPastRevenue frame is a standard content frame, allowing the thermometer component to be included in the report structure. It serves no other purpose.
- **Thermometer.** The thermoPastRevenue component is a thermometer, derived from the ssThermometer library component in the ssSmart.rol library. The Color, Label, and TriggerLabel properties are set in thermoPastRevenue to configure these features of the thermometer. The OnRow method (in code located both in thermoPastRevenue and in its parent, ssThermometer) retrieves the one row generated by the datastream and passes the PastRevenue value as the method's input parameter, DataValue. The other input parameters, MinimumValue, MaximumValue, and TriggerDataValue, are obtained from previous logic, as described in "Obtaining the Minimum, Maximum, and Trigger Values for Smart Reports" on page 157.

  An additional role of the OnRow method is to compare DataValue with TriggerDataValue in order to obtain a TRUE or FALSE value for PastRevenueThermometerAboveTarget. This Boolean variable and corresponding ones for the other thermometers are collectively used to determine the direction of the order-of-merit indicator arrow at the top of the account's page.

About Positioning a Thermometer on the Dashboard Subpage of a Smart Report

A *page* is a component that specifies the visual design of a page in the report. A page component primarily consists of flows, which determine the printable area of the page and may also contain graphics, labels, and various controls. When you run a report, Actuate builds frames to display the data. As each frame is created, Actuate places it in the flow on the current page, starting from the top of the flow and aligning each frame with the left edge of the flow. If a frame is too long to fit in the current flow, Actuate places it at the top of the next flow, which may be on the same page or another page. The main page layout for a Siebel report is specified in the PageStyle child component of the PageList component.

Every section in a report design can optionally have an associated *subpage*. A subpage is just like a page, except that it fits into a flow on the active page. The active page is the page in the report currently being generated when the subpage starts. Like a page, a subpage contains one or more flows and can contain frames and controls.

You usually use a subpage when you want to change the page style for the contents of a section in the middle of a page. In the case of Smart Reports, a subpage is used to position thermometers, graphics, and report text within the dashboard. In a parallel section such as the one used to implement the dashboard, each flow in the subpage corresponds to one of the report sections in the parallel section. This correspondence is implemented by entering the name of the flow in the FlowName property of each report section component in the parallel section. Normally this property is not used except to pair up report sections and subpage flows in a parallel section.

Figure 28 illustrates the report sections and subpage flows in the dashboard parallel section in the Account Summary report.

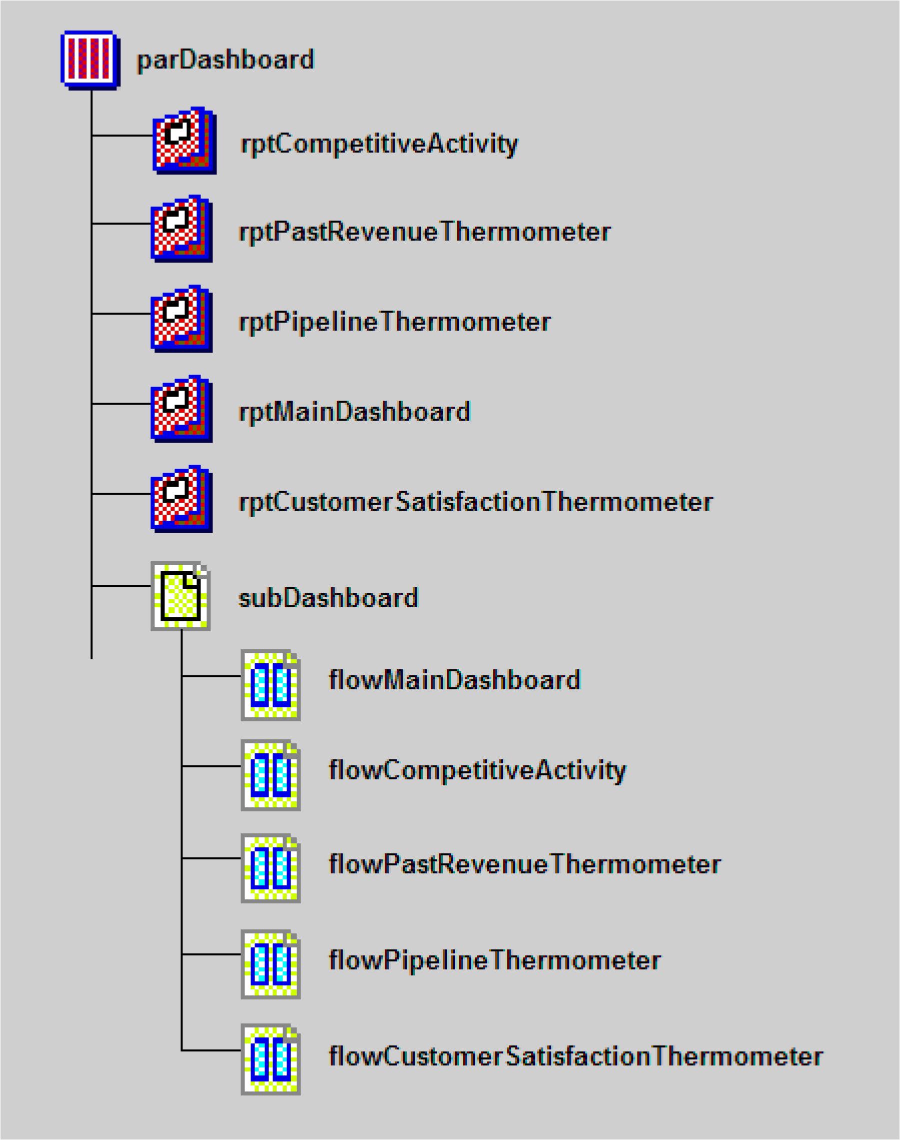


Figure 28. Dashboard Parallel Section Structure

The standard subpage and flow components are used, rather than specialized ssSmart.rol components. A subpage component is dragged from the Pages toolbar to the section to which it will be added. You then manually resize it by dragging the sizing handles. A flow is dragged from the Pages toolbar onto the subpage area and then resized and positioned. The size of the flow should be sufficient to hold the thermometer, the graph, the report text, or whatever is to be included, and any associated labels. Also, the flows should not overlap, although the text report flow in standard Smart Reports is often the size of the entire subpage for optimal use of space.

Passing Siebel-Generated Graphics to a Smart Report

Some specialized graphics in Smart Reports are too complex to be defined using AFC and sscustom graph classes. These must be generated in the Siebel application and transferred to the Actuate report as bitmaps by way of OLE automation (or similar application-to-application object communication). Some examples of passed graphics in Smart Reports are the three charts in the Account Service Detail Report: Severity Distribution, Service Request Substatus, and Aging By Severity.

Because creation and modification of specialized classes in Siebel Tools are not supported, Siebel Systems does not support creation and modification of passed graphics in Smart Reports by customers. If you have a requirement of this kind, contact Siebel Technical Services.

Designing a Smart Report

This section will explain how the Smart Reports are designed beginning with the Smart Report - Opportunity Detail report.

About the Opportunity Detail Smart Report

The Opportunity Detail report presents each opportunity with graphical information including order-of-merit indicator, Competitive Activity, Deal Size Thermometer, Buying Influencers Thermometer, Probability Thermometer, Sales Stage Slider, Contacts/Influence Map, and other details in the dashboard section and detail sections on contacts, products, competitors, activities, and notes. First a functional description of the components in this report is provided, followed by the report design in Actuate e.Report Designer Professional.

About the Functional Detail Section for the Opportunity Detail Smart Report

The functional detail includes a high-level description of all the components of the Opportunity Detail report. This section also describes the properties of the specialized graphical components.

About the Graphical Components for the Opportunity Detail Smart Report

Order-of-merit indicator. The order-of-merit indicator indicates the overall measure for the opportunity and is calculated based on the following logic:

All three Thermometers above target = Up

Two of the Three Thermometers above target = Right

Otherwise = Down

Sales stage slider. This indicates how close the opportunity is in terms of closing the deal.

Deal size thermometer. This thermometer displays the deal size of the opportunity relative to the average across all opportunities. The minimum value is taken as zero, the trigger value is calculated to be the average across all opportunities, and the maximum is twice the trigger value.

Competitive activity. The competitive activity depicts the top four competitors in descending order; the value indicates the relative threat, which is determined by the sales representative or the status of each competitor. The value displayed in the graph is obtained from the List of Values table under type = 'TARGET_COMP_THREAT' and Value = Threat.

Buying influencers thermometer. The Buying Influencers thermometer depicts the buying influence based on the weighted average of the contacts involved with the opportunity (TASOrgStatusArray, ContactRoleArray, and TASPolitical AnalysisArray).

Probability thermometer. This thermometer shows an amount between 0 and 100, the percentage probability of converting the opportunity into an order.

Contacts influence map. The map shows the contacts associated with the opportunity, highlighting the decision makers.

About Other Components of the Opportunity Detail Smart Report

Page header. The page header includes a snapshot of the opportunity and shows the name of the opportunity, the revenue, the probability, and the order-of-merit indicator.

Dashboard. In this section, the graphical and textual information for the opportunity is displayed. The thermometers described earlier are all in the dashboard, along with opportunity summary text.

Contact detail. This is one of the detail sections (or a subreport section) of the opportunity master section. The details of the contacts associated with this opportunity are shown in the form of a list report.

Products. The product offerings relevant to the opportunity are listed in this detail section.

Competitors. This section lists all the competitors for this opportunity.

Decision issues. In this section, the decision issues for the opportunity are listed in order of priority.

All activities. This section lists all the activities undertaken for this opportunity.

Notes. This section shows the email messages, correspondence, and proposals sent for this opportunity.

About Technical Detail in the Opportunity Detail Smart Report

The Opportunity Detail report consists of two report sections. The first is the data collection section and the second is the main section, as shown in Figure 29.

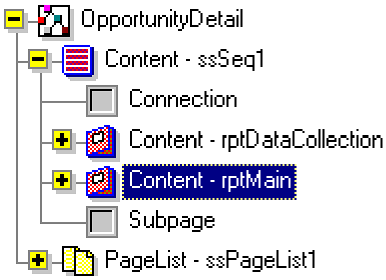


Figure 29. Opportunity Detail Report Main Section

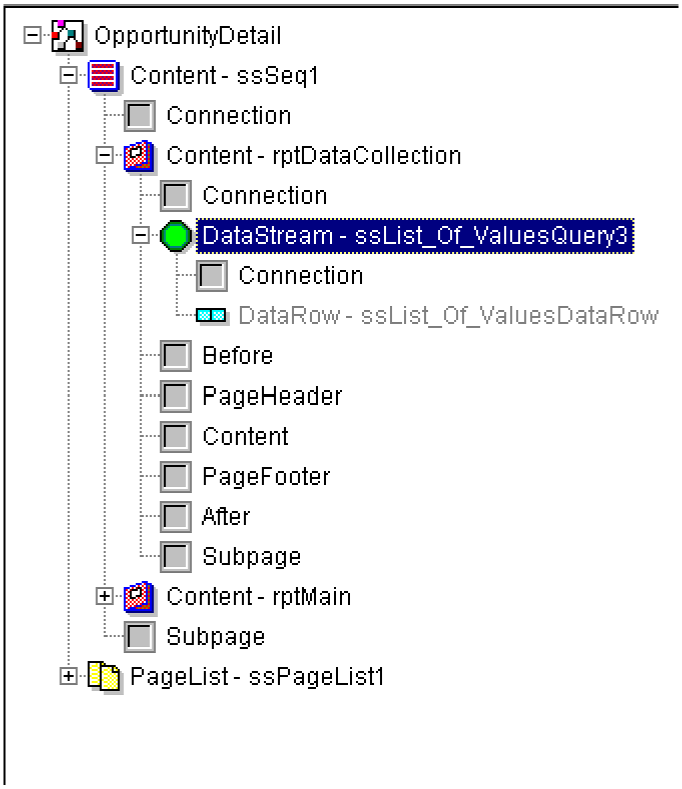
The purpose of the data collection section is to obtain and manipulate data for the current master record (which is a record from the Opportunity business component). Specifically, the data from the List_of_Values of Opportunity object is obtained in the Fetch method for determining the values for the Buying Influencers thermometer.

About the Data Collection Section of the Opportunity Detail Smart Report

List of Values data collection is required for weighting factors used in the Buying Influencers thermometer. Three types of weighting factors, CONTACT_ROLE, TAS_POLITICAL_ANALYSIS, and TAS_ORG_STATUS, are stored in separate static arrays defined in the OpportunityDetail component. Also, minimum and maximum weighting factors are stored in separate static variables for each of the three types.

Data rows are collected by a component subclassed from the ssList_Of_ValuesQuery class defined in ssSmart.rol. The SearchSpec property is set to [Type] = 'CONTACT_ROLE' or [Type] = 'TAS_POLITICAL_ANALYSIS' or [Type] = 'TAS_ORG_STATUS'; DefaultWeightingFactor is a local property set to 1; NeutralTAS_ORG_STATUS is a local property set to Neutral. The Fetch method of ssList_Of_ValuesQuery3 decodes the type of weighting factor and populates the variables described above.

The data collection section can be created using the following procedure. The contents of this section appear in Actuate e.Report Designer Professional, as shown in Figure 30 and explained in the procedure.



```
⊟ OpportunityDetail
    ⊟ Content - ssSeq1
        Connection
        ⊟ Content - rptDataCollection
            Connection
            ⊟ DataStream - ssList_Of_ValuesQuery3
                Connection
                DataRow - ssList_Of_ValuesDataRow
            Before
            PageHeader
            Content
            PageFooter
            After
            Subpage
        ⊞ Content - rptMain
        Subpage
    ⊞ PageList - ssPageList1
```

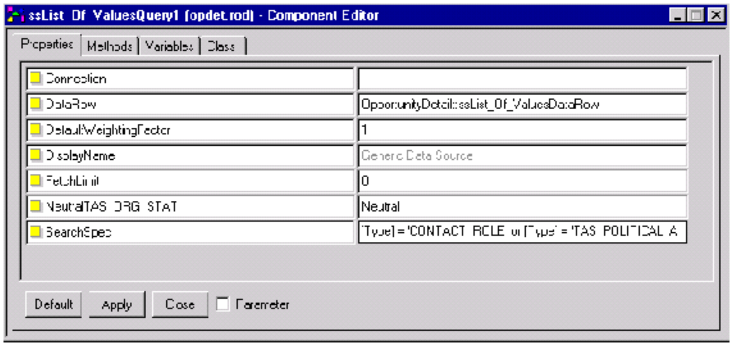
Figure 30. Data Collection Section

To create a data collection section for the Opportunity Detail Smart Report

1. Drag and drop a reference to the ssRpt class from the sscustom.rol into the Content slot of ssSeq1.
2. Right-click the reference and subclass it following the instructions in "To create a report design file in Actuate e.Report Designer Professional" on page 62.
3. Right-click again and choose Rename from the menu. Then type `rptDataCollection` in the box.
4. Drag and drop the Query button from the main toolbar into the DataStream component of the report section.
5. Click the library button in the menu bar and then double-click ssSmart.rol library.
6. Drag and drop a reference to the ssList_Of_ValuesQuery class from the ssSmart.rol library into the DataStream slot of rptDataCollection.
7. Subclass this component and rename it ssList_Of_ValuesQuery3.
8. Open the Component Editor window for ssList_Of_ValuesQuery3 by double-clicking.

9 Click the Variables tab and add two new properties: DefaultWeightingFactor (type=Integer) and NeutralTAS_ORG_STATUS (type=String).

10 Under the Properties tab set the DefaultWeightingFactor property to 1, set the NeutralTAS_ORG_STATUS property to Neutral, and set the SearchSpec to [Type] ='CONTACT_ROLE' or [Type] = 'TAS_POLITICAL_ANALYSIS' or [Type] = 'TAS_ORG_STATUS’. The resulting Component Editor Properties tab window is illustrated in the following figure*.*



Connection	
DataRow	OpportunityDetail::ssList_Of_ValuesDataRow
DefaultWeightingFactor	1
DisplayName	Generic Data Source
FetchLimit	0
NeutralTAS_ORG_STAT	Neutral
SearchSpec	[Type] = 'CONTACT_ROLE' or [Type] = 'TAS_POLITICAL_A

11 Click the Methods tab (shown in the figure following Step 10), choose the Fetch method and click Override. Include the following Actuate VB script in the Fetch method. This correctly changes the Fetch method from a Sub to a Function.

```
Function Fetch( ) As AcDataRow

    Dim aRow As ssList_Of_ValuesDataRow

        Set aRow = Super::Fetch( )

        Do While Not aRow Is Nothing

            If Trim$(aRow.ssWeighting_Factor) = "" Then aRow.ssWeighting_Factor =
        Str$(DefaultWeightingFactor)

            If aRow.ssType = "CONTACT_ROLE" Then

                ContactRoleArraySize = ContactRoleArraySize + 1

                If MaxContactRole < Val( aRow.ssWeighting_Factor ) Then MaxContactRole
        = Val( aRow.ssWeighting_Factor )

                If MinContactRole > Val(aRow.ssWeighting_Factor) Then MinContactRole =
        Val(aRow.ssWeighting_Factor)

                ContactRoleArray( 2, ContactRoleArraySize ) = aRow.ssWeighting_Factor

                ContactRoleArray( 1, ContactRoleArraySize ) = aRow.ssName

        ElseIf aRow.ssType = "TAS_POLITICAL_ANALYSIS" Then

                TASPoliticalAnalysisArraySize = TASPoliticalAnalysisArraySize + 1

                If MaxTASPoliticalAnalysis < Val( aRow.ssWeighting_Factor Then
        MaxTASPoliticalAnalysis = Val(aRow.ssWeighting_Factor)
```

```
            If MinTASPoliticalAnalysis > Val(aRow.ssWeighting_Factor) Then
      MinTASPoliticalAnalysis = Val(aRow.ssWeighting_Factor)

            TASPoliticalAnalysisArray( 2, TASPoliticalAnalysisArraySize =
      aRow.ssWeighting_Factor

            TASPoliticalAnalysisArray( 1, TASPoliticalAnalysisArraySize =
      aRow.ssName

      ElseIf aRow.ssType = "TAS_ORG_STATUS" Then

            TASOrgStatusArraySize = TASOrgStatusArraySize + 1

            If MaxTASOrgStatus < Val( aRow.ssWeighting_Factor ) Then MaxTASOrgStatus
      = Val( aRow.ssWeighting_Factor )

            If MinTASOrgStatus > Val(aRow.ssWeighting_Factor) Then MinTASOrgStatus
      = Val(aRow.ssWeighting_Factor)

            If aRow.ssName = NeutralTAS_ORG_STATUS Then NeutralWeightingFactor =
      Val(aRow.ssWeighting_Factor)

            TASOrgStatusArray( 2, TASOrgStatusArraySize ) = aRow.ssWeighting_Factor

            TASOrgStatusArray( 1, TASOrgStatusArraySize ) = aRow.ssName

      Else

            ssDisplayMessage("Invalid LOV Type" & aRow.ssType )

      End If

            Set aRow = Super::Fetch( )

   Loop

   Set Fetch = Nothing

End Function
```

This method requires variables to be defined in the OpportunityDetail report component (the topmost component) as shown in Table 13.

Table 13. OpportunityDetail Report Components

Variable Name	Type	Storage	Visibility	Comment
ContactRoleArray(2,10)	String	Static	Public	Stores contact roles and weighting factors
ContactRoleArraySize	Integer	Static	Public	Used in array integration loops
TASOrgStatusArray(2,10)	String	Static	Public	Stores organization status identifiers and weighing factors

Table 13. OpportunityDetail Report Components

Variable Name	Type	Storage	Visibility	Comment
TASOrgStatusArraySize	Integer	Static	Public	Used in array integration loops
TASPoliticalAnalysisArray (2.10)	String	Static	Public	Stores political analysis identifiers and weighing factors
TASPoliticalAnalysisSize	Integer	Static	Public	Used in array integration loops
MaxContactRole	Integer	Static	Public	Maximum weighting factor
MinContactRole	Integer	Static	Public	Minimum weighting factor
MaxTASOrgStatus	Integer	Static	Public	Maximum weighting factor
MinTASOrgStatus	Integer	Static	Public	Minimum weighting factor
MaxTASPoliticalAnalysis	Integer	Static	Public	Maximum weighting factor
MinTASPolitical Analysis	Integer	Static	Public	Minimum weighting factor

About the Main Report Section of the Opportunity Detail Smart Report

The Main Report section provides the structure for gathering and displaying all the data for each opportunity available from the Opportunity business component. Making the Opportunity data row available to detail sections (or subreports) requires static data row storage. OpportunityRow is a static variable of type ssOpportunityDataRow, defined in the OpportunityDetail component. The OnRow method of the rptMain report section assigns this variable equal to the current row from the ssOpportunityQuery as shown below.

```
Sub OnRow( row As AcDataRow )
   Super::OnRow( row )
   Set OpportunityRow = row
End Sub
```

The Main Report section contains components in the DataSource, PageHeader, and Content slots, as shown in Figure 31.

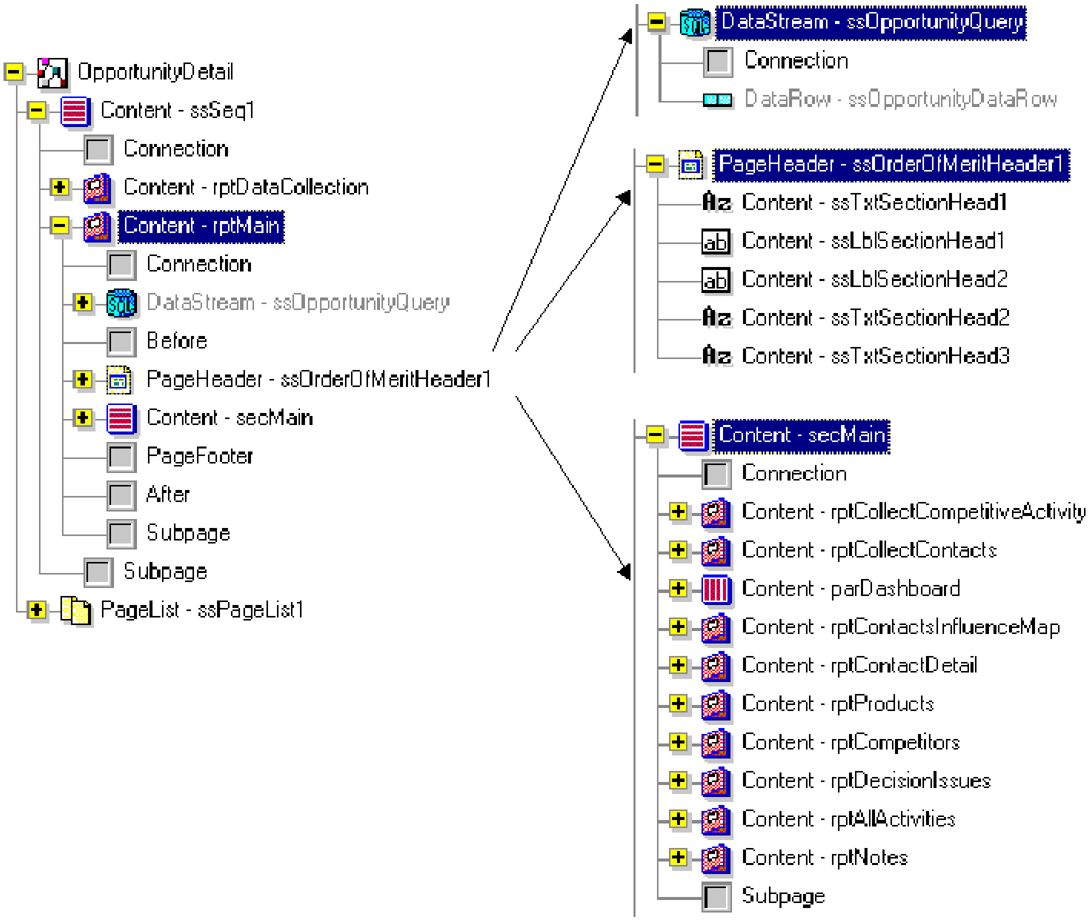

OpportunityDetail
Content - ssSeq1
Connection
Content - rptDataCollection
Content - rptMain
Connection
DataStream - ssOpportunityQuery
Before
PageHeader - ssOrderOfMeritHeader1
Content - secMain
PageFooter
After
Subpage
Subpage
PageList - ssPageList1
DataStream - ssOpportunityQuery
Connection
DataRow - ssOpportunityDataRow
PageHeader - ssOrderOfMeritHeader1
Content - ssTxtSectionHead1
Content - ssLblSectionHead1
Content - ssLblSectionHead2
Content - ssTxtSectionHead2
Content - ssTxtSectionHead3
Content - secMain
Connection
Content - rptCollectCompetitiveActivity
Content - rptCollectContacts
Content - parDashboard
Content - rptContactsInfluenceMap
Content - rptContactDetail
Content - rptProducts
Content - rptCompetitors
Content - rptDecisionIssues
Content - rptAllActivities
Content - rptNotes
Subpage

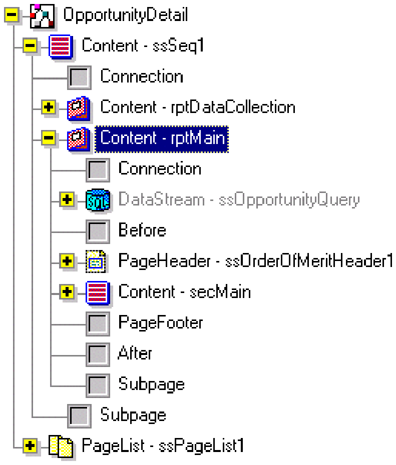
Figure 31. Main Report Section

- The DataStream section obtains data from the main business component, Opportunity, for this report.
- PageHeader contains the information displayed in the header section of the report: the name of the opportunity, the revenue, the probability, and the order-of-merit indicator.
- The Main Content section consists of the components (data collection sections, dashboard parallel section, and detail sections).
    - The Data Collection section consists of the Competitive Activity and Contacts sections. (The Data Collection sections are needed because this data is required in multiple report sections for each opportunity, but it is only possible to query the linked subreport business components once for each opportunity.)

- The Dashboard parallel section contains sales stage slider graphics, a Main Dashboard section, a Competitive Activity section, a Deal Size thermometer, a Buying Influencers thermometer, and a Probability thermometer.
- The Detail section consists of subsections on the contacts influence map, contact detail, products, competitors, decision issues, activities, and notes.

Creating the Sequential Report Section in the Opportunity Detail Smart Report

Create the main report section by subclassing from ssRpt into the content slot of ssSeq1. Rename this Report Section to rptMain. The rptMain section appears in Actuate e.Report Designer Professional as shown in Figure 32.



```
⊟ OpportunityDetail
  ⊟ Content - ssSeq1
      Connection
    ⊞ Content - rptDataCollection
    ⊟ Content - rptMain
        Connection
      ⊞ DataStream - ssOpportunityQuery
        Before
      ⊞ PageHeader - ssOrderOfMeritHeader1
      ⊞ Content - secMain
        PageFooter
        After
        Subpage
      Subpage
  ⊞ PageList - ssPageList1
```

Figure 32. rptMain Section

Defining the DataStream Section in the Opportunity Detail Smart Report

Click the library button and double-click opdet.rol to open the library. Drag and drop ssOpportunityQuery into the DataStream section under rptMain. It is not necessary to subclass.

Defining the Page Header Section in the Opportunity Detail Smart Report

The PageHeader frame provides the Order Of Merit graphic display because it is subclassed from the ssOrderOfMeritHeader class defined in ssSmart.rol library. The direction of the order-of-merit indicator is determined by assigning the ArrowDirection class variable in the Finish method before calling the superclass Finish method. In this method, the arrow direction is determined by the variable NumberOfThermometersAboveTarget, which is a static integer variable defined in the OpportunityDetail component.

If all three thermometers display data values higher than their trigger values, the up arrow is displayed. If only two of the thermometer data values exceed their trigger values, the right arrow is displayed. If one or none of the thermometer data values exceeds its trigger value, the down arrow is displayed.

To create a Page Header section in the Opportunity Detail Smart Report

1 Click the library button and double-click ssSmart.rol to open the library. Drag and drop the ssOrderOfMeritHeader frame into the Page Header section under rptMain. Subclass this frame and rename it ssOrderOfMeritHeader1.

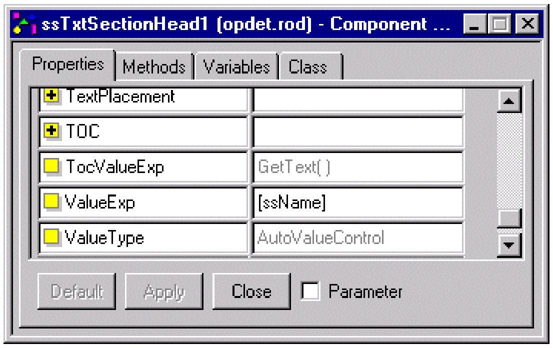
2 Override the Finish method as follows:

```
Sub Finish( )

    If NumberOfThermometersAboveTarget = 3 Then

        ArrowDirection = "UP"

    ElseIf NumberOfThermometersAboveTarget < 2 Then

        ArrowDirection = "DOWN"

    Else

        ArrowDirection = "PUSH"

        End If

    Super::Finish( )

End Sub
```

3 Define the NumberOfThermometersAboveTarget variable (Type=Integer, Storage=Static) in the OpportunityDetail component.

4 Create the contents of the Page Header section by dragging and dropping the appropriate text control and label controls. First click the library button and double-click sscustom.rol to open the library.

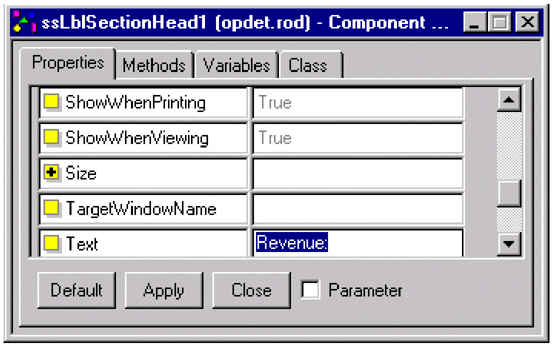
5 Drag and drop the ssTxt text control from this Library Browser into the ssOrderOfMeritHeader1 frame. Rename the ssTxt control ssTxtSectionHead1. Then highlight ssTxtSectionHead1, right-click, and choose Properties from the menu. Select the value in ValueExp to be the [ssName] variable as reflected in the following figure.



TextPlacement	
TOC	
TocValueExp	GetText( )
ValueExp	[ssName]
ValueType	AutoValueControl

6 Repeat Step 5 two more times and rename the ssTxt control ssTxtSectionHead2 and ssTxtSectionHead3, respectively. Select the ValueExp in these text controls to be [ssRevenue_Formatted] and [ssRep__] & "%", respectively.

7 Drag and drop the ssLblBlueBlack label control from the sscustom.rol Library Browser into the ssOrderOfMeritHeader1 frame. Rename the label control ssLblSectionHead1. Then select this label, right-click and choose Properties from the menu. In the Properties tab, enter `Revenue:` in the field against Text as shown in the following figure.



ShowWhenPrinting	True
ShowWhenViewing	True
Size	
TargetWindowName	
Text	Revenue:

8 Repeat Step 7 and rename the label ssLblSectionHead2. Enter `Probability:` in the field against Text Property.

Defining the Content Main Section in the Opportunity Detail Smart Report

Subclass from ssSeq to create secMain in the Content slot of rptMain.

About the rptCollectCompetitiveActivity Section in the Opportunity Detail Smart Report

The rptCompetitiveActivity and rptCompetitors sections require Competitor data collection. The results of ssCompetitorQuery_3 query in Opdet.rol are stored in CompetitorDataList, a static variable defined in the OpportunityDetail component.

CompetitorIndexArray, an array of list position numbers, is maintained to facilitate retrieval of the four competitors with the highest threat value. The array is defined in the OpportunityDetail component.

The sifCompetitiveActivity component is subclasssed from the ssSingleInputFilter class in sscustom. The Start method of the sifCompetitiveActivity filter initializes the array values to zero for each opportunity. The Start method of the rptCollectCompetitiveActivity report section initializes the list for each opportunity. The Fetch method of the filter populates the list and the array.

To create a rptCollectCompetitiveActivity section in the Opportunity Detail Smart Report

1 Place, subclass, and rename the components for the rptCollectCompetitiveActivity section.

2 Define the CompetitorDataList variable (Type=AcList, Storage=Static) in the OpportunityDetail component.

3 Override the Start method in sifCompetitiveActivity to initialize the CompetitorIndexArray for each opportunity.

```
Function Start( ) As Boolean
```

```
    Dim i As Integer

Start = Super::Start( )

    For i = 0 To 3

        CompetitorIndexArray(i) = 0

    Next

End Function
```

4 Override the Fetch method in sifCompetitiveActivity to obtain the top four competitors and the associated threat values as entered by the sales representative and to build the CompetitorDataList.

```
Function Fetch( ) As AcDataRow

    Dim aRow As ssCompetitorDataRow

    Dim bRow As ssCompetitorDataRow

    Dim i As Integer

    Dim j As Integer

    Set aRow = New ssCompetitorDataRow

    Set aRow = InputAdapter.Fetch()

    If Not aRow Is Nothing Then

        CompetitorDataList.AddToTail(aRow)

        For i = 0 To 3

                If CompetitorIndexArray(i) > 0 Then

                Set bRow = CompetitorDataList.GetAt(CompetitorIndexArray(i))

                If Val(aRow.ssThreat_Value) >= Val(bRow.ssThreat_Value) Then

                For j = 3 T o i Step - 1

                CompetitorIndexArray(j + 1) = CompetitorIndexArray(j)

                Next

                CompetitorIndexArray(i) = CompetitorDataList.GetCount()

                Exit For

            End If

        Else

            CompetitorIndexArray(i) = CompetitorDataList.GetCount()

            Exit For
        End If
    Next
    Set Fetch = aRow
    End If
End Function
```

About the rptCollectContacts Section in the Opportunity Detail Smart Report

The rptBuyingInfluencersThermometer, rptContactsInfluenceMap and rptContactDetail sections require Contact data collection. The results of ssContactQuery_1 query defined in Opdet.rol, are stored in the ContactDataList, a static variable defined in the OpportunityDetail component.

To create a rptCollectContacts section in the Opportunity Detail Smart Report

1. Place, subclass, and rename the components for the rptCollectContacts section.
2. Define the ContactDataList variable (Type=AcList, Storage=Static) in the OpportunityDetail component.
3. Include code under the Fetch method in ftrContacts to obtain the contacts list.

```
    Function Fetch( ) As AcDataRow
        Dim aRow As ssContactDataRow
            Do
                Set aRow = InputAdapter.Fetch()
                If aRow Is Nothing Then Exit Function
                ContactDataList.AddToTail(aRow)
        Loop
End Function
```

About the parDashBoard Section of the Opportunity Detail Smart Report

See the general Smart Reports documentation for a general description of DashBoard parallel sections ("About the Report Structure for Smart Reports" on page 152).

To create a parDashBoard section in a Smart Report

1. Subclass from ssParallelSection to create parDashBoard in the Content slot of secMain.
2. Set up subDashBoard in the SubPage slot.

Defining the Subpage Layout Section in the Opportunity Detail Smart Report

In the subpage layout, set up a flow for each of the report sections described in the sections that follow.

rptSalesStageSlider. The frmStageSlider component simulates a horizontal bar graph by dynamically instantiating a dark-background control and sizing it based on the value of the Sales Stage Win Percent field for the current opportunity. Other elements of the graphic, such as ticks and label, are placed in the frmStageSliderHolder frame.

rptMainDashboard. The MainDashboard report section displays details for this opportunity and a general functional description of this report.

rptDealSizeThermometer. The DealSizeThermometer component is a subclass of the ssThermometer defined in ssSmart.rol. The four thermometer control variables, TriggerDataValue, DataValue, MinimumValue, and MaximumValue, are set in the OnRow method. TriggerDataValue is set from the Order By LOV field, MaximumValue is set from the Target High LOV field, and MinimumValue is set from the Target Low LOV field. DataValue is set from the Functional Revenue field of OpportunityRow, which was stored earlier. The OnRow method also adjusts the NumberOfThermometersAboveTarget variable to have the appropriate effect on the OrderOfMerit image.

rptCompetitiveActivity. The Competitive Activity chart is a standard Actuate horizontal-bar-chart summary graph. The data rows that feed the graph are pulled from the CompetitorDataList using the list row numbers saved in CompetitorIndexArray. The list and the array were saved earlier in the rptCollectCompetitiveActivity report section. The number of rows fed to the graph is limited to a maximum of four, and they are ordered so that the most threatening competitor is shown at the top.

rptBuyingInfluencersThermometer. The BuyingInfluencersThermometer graphic display is driven by scores calculated from values stored in arrays by the rptDataCollection section.

The ssContactQuery_2 data source component iterates through the ContactDataList, feeding the data rows to the ftrCalculateWeightedScore single input filter. The Fetch method for ssContactQuery_2 follows.

```
Function Fetch( ) As AcDataRow

    Dim aRow As ssContactDataRow

        If Position <= ContactDataList.GetCount() Then

            Set aRow = ContactDataList.GetAt(Position)

            Set Fetch = aRow

            AddRow(Fetch)

        End If

End Function
```

The Fetch method of the ftrCalculateWeightedScore single input filter calculates a score for an opportunity by averaging the scores for each contact associated with that opportunity. The score for each contact is the product of three weighting factors collected earlier and stored in static arrays.

The first weighting factor is retrieved from the TASOrgStatusArray by matching the ‘Org Status’ field with a TAS_ORG_STATUS type LOV value stored in the array. The second weighting factor is retrieved from the ContactRoleArray by matching the ‘Role’ field with a CONTACT_ROLE type LOV value stored in the array. The third weighting factor is retrieved from the TASPoliticalAnalysisArray by matching the ‘Political Analysis’ field with a TAS_POLITICAL_ANALYSIS type LOV value stored in the array. The script for ftrCalculateWeightedScore follows.

```
Function Fetch( ) As AcDataRow

   Dim aRow As ssContactDataRow

   Dim finalRow As OpportunityDetail::rowBuyingInfluencers

   Dim i As Integer

   Dim score As Integer

   Dim totalScore As Integer

   Dim numberOfContacts As Integer

   Set aRow = Super::Fetch( )

   If aRow Is Nothing AND Position > 1 Then Exit Function

   Do While Not aRow Is Nothing

       If Not Trim$(aRow.ssOrg_Status) = "" Then

          For i = 1 to TASOrgStatusArraySize

          If aRow.ssOrg_Status = TASOrgStatusArray( 1, i ) Then

                score = Val( TASOrgStatusArray( 2, i ) )

                Exit For

          End If

       Next

       For i = 1 to ContactRoleArraySize

          If aRow.ssRole = ContactRoleArray( 1, i ) Then score =

                score * Val( ContactRoleArray( 2, i ) )

                Exit For

          End If

       Next

       For i = 1 To TASPoliticalAnalysisArraySize

          If aRow.ssPolitical_Analysis =

                TASPoliticalAnalysisArray1, i ) Thenscore = score *
                Val( TASPoliticalAnalysisArray( 2, i ) )
                Exit For
          End If
       Next
       totalScore = totalScore + score
       numberOfContacts = numberOfContacts + 1
    End If
    Set aRow = Super::Fetch( )
   Loop
   Set finalRow = New OpportunityDetail::rowBuyingInfluencers
   If numberOfContacts = 0 Then
      finalRow.Score = 0
   Else
      finalRow.Score = totalScore / numberOfContacts
   End If
   Set Fetch = finalRow
   AddRow( Fetch )
End Function
```

rptProbabilityThermometer. The ProbabilityThermometer component is a subclass of ssThermometer, defined in ssSmart.rol. As usual, the four thermometer control variables, TriggerDataValue, DataValue, MinimumValue, and MaximumValue, are set in the OnRow method.

TriggerDataValue is set from the Order By LOV field, MaximumValue is set from the Target High LOV field, and MinimumValue is set from the Target Low LOV field. DataValue is set from the Rep% field for the opportunity. The OnRow method also adjusts the NumberOfThermometersAboveTarget variable so it will have the appropriate effect on the OrderOfMerit image. The script for the OnRow method follows.

```
Sub OnRow( row As AcDataRow )
   Dim aRow As ssList_Of_ValuesDataRow
Super::OnRow( row )
   Set aRow = row
   DataValue = Val( OpportunityRow.ssRep__ )
   TriggerDataValue = Val( aRow.ssOrder_By )
```

```
        MaximumValue = Val( aRow.ssTarget_High )

        MinimumValue = Val( aRow.ssTarget_Low )

        If DataValue > TriggerDataValue Then

            NumberOfThermometersAboveTarget = NumberOfThermometersAboveTarget + 1

        End If

End Sub
```

rptContactDetail. Instructions on creating the rptContactDetail section follow.

To create a rptContactDetail section in the Opportunity Detail Smart Report

1. Define the rptContactDetail report section, add a Query DataStream, and rename it dsTransferContactList.

    a. In Properties, make DataRow ssContactDataRow.

    b. Add a Before section by dragging and dropping a frame from the main toolbar into the rptContactDetail section.

       Similarly, define the PageHeader and Content sections.

       The Before section includes the labels in the header for the Contacts child section.

    c. Drag and drop the ssLblSectionHead control from sscustom.rol into the frmContactsHeadings1 frame to create the headings for the Contacts child section.

       These include lblName, lblJobTitle, lblAccount, lblSite, lblWorkPhone, lblWorkFax, and lblRole, and correspond to the titles Name, Job Title, Account, Site, Work Phone, Work Fax, and Role.

2. In the PageHeader section (frmContactsContinuedHeader frame), drag and drop the ssLabelSectionHeadContinued control from sscustom.rol to create lblContacts and lblContinued titles.

   Also, drag and drop ssLineControlP from sscustom.rol and rename it LineControlP2.

3. In the content frame (frmMainContactData), drag and drop lblBlueBlack control from sscustom.rol to create the contents lblName, lblJobTitle, lblAccount, lblSite, lblWorkPhone, lblWorkFax, and lblRole, which correspond to the contents Name, Job Title, Account, Site, Work Phone, Work Fax, and Role, respectively.

   These appear as list columns under the child report section Contacts and get values from ssFull_Name, ssJob_Title, ssssAccount, ssAccount_Location, ssWork_Phone_, ssFax_Phone__, and ssRole, respectively, by setting the ValueExp property.

rptProducts. Instructions on creating the rptProducts section follow.

To create a rptProducts section in the Opportunity Detail Smart Report

1. Define the rptProducts report section and drag and drop the ssOpportunity_ProductQuery2 data source from the opdet.rol library.

2. Drag and drop a frame into the Page Header section and rename it frmHeaderAndTitleforProducts, and include the necessary control as part of the Page Header for the Products child report. This includes the headings Product, Expected Delivery Date, Quantity, and Comment.
3. Include the Contents section to appear as list columns under the child report section Products and get values from ssVendor, ssStatus, and ssComment by setting the ValueExp property.

rptCompetitors section. Instructions on creating the rptCompetitors section follow.

To create a rptCompetitors section in the Opportunity Detail Smart Report

1. Define the rptCompetitors report section and drag and drop the ssCompetitorQuery_3 data source from the opdet.rol library.
2. Drag and drop a frame into the PageHeader section and rename it frmHeaderAndTitleforCompetitors1. Include the necessary control as part of the Page Header for the Competitors child report section.
3. Include the Contents section to appear as list columns under the child report section Competitors and get values from ssProduct, ssExpectedDeliveryDate, ssQuantity, and ssComment from ValueExp. Status, and ssComment by setting the ValueExp property.

rptDecisionIssues section. Instructions on creating the rptDecisionIssues section follow.

To create a rptDecisionIssues section in the Opportunity Detail Smart Report

1. Define the rptDecisionIssues report section and drag and drop the ssDecision_IssueQuery_4 data source from the opdet.rol library.
2. Drag and drop a frame into the PageHeader section and rename it frmHeaderAndTitleforDecisionIssues, and include the necessary control as part of the Page Header for the DecisionIssues child report section.
3. Include the Contents section to appear as list columns under the child report section Decision Issues and get values from ssName, ssComment, and the ssRank product by setting the ValueExp property.

rptAllActivities section. Instructions on creating the rptAllActivities follow.

To create a rptAllActivities section in the Opportunity Detail Smart Report

1. Define the rptAllActivities report section and drag and drop the ssActionQuery_5 data source from the opdet.rol library.
2. Drag and drop a frame into the PageHeader section and rename it frmHeaderAndTitleforAllActivities1. Include the necessary control as part of the Page Header for the All Activities child report section.
3. Include the Contents section to appear as list columns under the child report section All Activities and get values from ssFull_Name, ssStart_Date-Formatted, ssType, ssDescription, and ssStatus. Drag and drop the Red Dot control from ssSmart.rol and include the OnRow method to indicate incomplete activities in red.

rptNotes section. Instructions on creating the rptNotes section follow.

To create a rptNotes section in the Opportunity Detail Smart Report

1. Define the rptNotes report section and drag and drop the ssOpportunity_NoteQuery_6 data source from the opdet.rol library.
2. Drag and drop a frame into the PageHeader section and rename it frmHeaderAndTitleforNotes1. Include the necessary control as part of the page header for the DecisionIssues child report section.
3. Include the Contents section to appear as list columns under the child report section Notes and get values from ssNote, ssCreated_By_Name, and ssCreated_Formatted by setting the ValueExp property.

About the Account Service Detail Smart Report

The report summarizes all the service-related information pertaining to the account. It contains information about currently open service requests, customer satisfaction, and the historical service request resolution for the account.

Here, the specialized graphical components and the related sections (including the data collection section) in Account Service Detail report are described. Except for specialized graphical elements and the related data collection sections, the contents of all Smart Reports are very similar and therefore are not described in remaining reports.

About the Order Of Merit Graphic in a Smart Report

The Order Of Merit graphic is determined in the Finish method of the ssOrderOfMeritHeader1 frame according to the following logic. Add values according to Table 14.

Table 14. Thermometer Variables for Order of Merit Graphic

Thermometer Variable	Value If Above Target (Trigger)
Customer Satisfaction	3
Open Service Request	2
Revenue	1

If the result is greater than 4, then the arrow direction is up. If the result is 3 or 4, then the arrow direction is facing right (push). Otherwise, the arrow direction is down.

About Data Collection and Calculation in the Account Service Detail Smart Report

All service requests for an Account are collected and stored in memory lists in the rptAllServiceRequests report section. Open Service Requests are stored in a separate list for use in the ensuing detail sections. Closure times for closed service requests are divided up into an array of lists that will be used in the rptClosureTimesBySeverity section to feed the line graphs, traffic lights, and calculated summary data.

As the arrays are populated, totals and counts are maintained to facilitate calculation of averages and standard deviations. High and low closure times are also stored for each severity, as determined by the Status field of the Service Request data row.

About the Dashboard Function in the Account Service Detail Smart Report

The thermometers in this dashboard function are like other Smart Report thermometers. They are, however, different in that they reside as frames within a single frame instead of in separate flows belonging to a subpage in a parallel section. The Start method of each thermometer sets the data, trigger, maximum, and minimum variables from report-level variables calculated or collected earlier in the report.

About Revenue Thermometer in the Account Service Detail Smart Report

Table 15 reflects the variables for use with the Revenue Thermometer.

Table 15. Variables for the Revenue Thermometer

Thermometer	Report-level Variable	Calculation Point	Comment
MaximumValue	revenueHigh	dsRevenueLOV	Collected once per report.
MinimumValue	revenueLow	dsRevenueLOV	Collected once per report.
TriggerDataValue	revenueAverage	dsRevenueLOV	Collected once per report.
DataValue	totalRevenue	sifAllOpportunities	The sum of the revenues for all opportunities associated with this account.

About the Open Service Request Thermometer in the Account Service Detail Smart Report

Table 16 reflects the variables for use with the Open Service Request Thermometer.

Table 16. Variables for the Open Service Request Thermometer

Thermometer Variable	Report-Level Variable	Calculation Point	Comment
MaximumValue	NA	Local	TriggerDataValue * 1.5.
MinimumValue	NA	Local	Fixed at zero.
TriggerDataValue	avgOpenSRs	dsTargetOpenSRsLOV	Collected once per report.
DataValue	countOpenSRs	sifAllServiceRequests	The count of open service requests for this account.

About the Customer Satisfaction Thermometer in the Account Service Detail Smart Report

Table 17 reflects the variables for use with the Customer Satisfaction Thermometer.

Table 17. Variables for the Customer Satisfaction Thermometer

Thermometer Variable	Report-level Variable	Calculation Point	Comment
MaximumValue	CustomersatisfactionHigh	ssList_Of_ValuesQuery1	Collected once per report.
MinimumVales	CustomerSatisfactionLow	ssList_Of_ValuesQuery1	Collected once per report.
TriggerDataValue	CustomerSatisfactionTarget	ssList_Of_ValuesQuery1	Collected once per report.
DataValue	TotalSatisfactionScore, countSurveys	local	The ratio of the total satisfaction score to the number of surveys.

About the rptAllServiceRequestsMethod in the Account Service Detail Smart Report

The method overrides of the sifAllServiceRequests (single input filter class) perform list storage, array storage, and calculations. Table 18 shows the methods that can be overridden for sifAllServiceRequests.

Table 18. Methods Overrides for sifAllServiceRequests

Method	Scope	Comment
Fetch	sisAllServiceRequests	Stores a list of data rows for open service requests. Calculates the closure time for closed service requests and stores them in an array of lists, one list for each severity. Accumulates total closure time and a count for each severity. Tracks high and low closure time for each severity and stores them in arrays.
Finish	sifAllServiceRequests	Calculates average closure time and standard deviation for each severity and stores the values in arrays.

About the Closure Time Summary Data Display Section in the Account Service Detail Smart Report

The following section describes the Closure Time Summary Data display.

About the rptClosureTimesAllSeverities Section in the Account Service Detail Smart Report

This is an outer report section that produces one blank data row for each Service Request Severity. This makes possible the reuse of rptClosureTimesBySeverity report section for a variable number of severities. Table 19 shows the relevant variables for rptClosureTimesAllSeverities.

Table 19. Relevant Variables for rptClosureTimesAllSeverities

Variable Name	Scope	Type	Storage	Visibility	Comment
MaxSeverity	AccountService Detail	Integer	Static	Public	Total number of possible service request severities.
currentSeverity	AccountService Detail	Integer	Static	Public	Used by the inner report section.

The Fetch method overrides to result in subreport executing once for each severity. Table 20 shows the scope for this method.

Table 20. Fetch Method Override

Method	Scope	Comment
Fetch	dsOneBlankRowPerSeverity	Produces one blank row per service request severity.

About the rptClosureTimesBySeverity Section in the Account Service Detail Smart Report

This is an inner report section that produces identically formatted closure time summary information for each service request severity. Service request data was collected earlier and stored in an array of lists. The data list used for an instance of this report section is specified by the currentSeverity variable. The components in this report section are described below.

dsGatherOneList. The code in dsGatherOneList class sorts the list by resolution time so that the medium closure time can be determined. The rows are then pulled from the list in correct sort order and passed to sifClosureTimes class. Table 21 explains how the Start and Fetch methods affects each class.

Table 21. Methods for dsGatherOneList

Method	Scope	Comment
Start	dsGatherOneList	Sorts the list of service requests by resolution time.
Fetch	dsGatherOneList	Pulls the service request data rows from the list in proper sort order.
Start	sifClosureTimes	Establishes graph horizontal boundaries as plus or minus 2 standard deviations from the mean. Establishes the median position in the data list.
Fetch	sifClosureTimes	Produces a data row for each of a predetermined number of buckets that correspond to a time increment. The time increments are represented by the x-axis of the graph. The count of service requests with closure times falling within the time increment is represented on the y-axis.

Before frame. Code in the Before frame produces and positions the graph target closure time marker. Table 22 reflects the method that needs changing to produce the closure time marker.

Table 22. Method to Produce the Closure Time Marker

Method	Scope	Comment
Finish	ssFrmP1	Dynamically produces the visual line element representing the closure time goal.

Summary data values. Summary data values are calculated in code. Table 23 reflects the method to use to calculate the summary data values.

Table 23. Methods for Calculating Summary Data Values

Method	Scope	Comment
Finish	txtSeverityLabel	From currentSeverity variable.
Finish	txtGoal	From targetResolutionTime array.
Finish	txtMedian	From medianResolutionTime array.
Finish	txtHigh	From highResolutionTime array.
Finish	txtLow	From lowResolutionTime array.
Finish	txtMean	From meanResolutionTime array.
Finish	tstStdDeviation	FromstdDevResolution Time array.
Finish	txtTotal	From countClosedSRs array.

Traffic lights. Traffic lights are resized, repositioned, and colored in code. Table 24 shows how code is used in methods to change Traffic lights.

Table 24. Methods Used for Traffic Lights

Method	Scope	Comment
Finish	dotTop	Green light. Mean closure time is faster than the target.
Finish	dotMiddle	Yellow light. Mean closure time is slower than the target, but not by more than one standard deviation.
FInish	dotBottom	Red light. Mean closure tine is slower than target by more than one standard deviation.

Closure time. Closure time graph y-axis labels are customized in code. Table 25 shows the method to use to produce better performance for the Closure Time class.

Table 25. Method Used for Closure Time

Method	Scope	Comment
Custom YLabels	ssSummaryGraph1	Produces improved graph performance over a wide range of sample sizes.

sifClosureTimes. Code in the sifClosureTimes class initializes the graph boundaries, establishes the sorted list position of the median closure time and counts service requests for each time increment, or bucket. One data row represents each bucket on the graph. Table 26 shows the variables for use with this class.

Table 26. Variables for Use With sifClosureTimes Class

Variable Name	Scope	Type	Storage	Visibility	Comment
graphMax	sifClosureTimes	Double	Static	Public	Closure time for right edge of graph.
graphMin	sifClosureTimes	Double	Static	Public	Closure time for left edge of graph.
meanResolutionTime()	AccountServiceDetail	Double	Static	Public	Array populated in sifAllServiceRequests.
stdDevResolutionTime()	AccountServiceDetail	Double	Static	Public	Array populated in sifAllServiceRequests.
targetResolutionTime()	AccountServiceDetail	Double	Static	Public	Set from LOV for each severity. **NOTE:** If modified, these LOVs should be maintained in the format dd:hh:mm.
targResTimeUnits()	AccountServiceDetail	Double	Static	Public	Used to scale summary data to hours, days, or weeks.
targResTimeGraphPercent	AccountServiceDetail	Double	Static	Public	Scales position of target indicator.

Table 26. Variables for Use With sifClosureTimes Class

Variable Name	Scope	Type	Storage	Visibility	Comment
currentSeverity	AccountServiceDetail	Integer	Static	Public	Set by outer report section. Used as an index into all arrays sized to MaxSeverities.
maxBuckets	sifClosureTimes	Integer	Static	Public	Determines horizontal resolution of graphs.
bucketIncrement	sifClosureTimes	Double	Static	Public	Width in minutes of closure time bucket.
ResolutionTimeList()	sifLCsureTimes	AcList	Static	Public	Array of data lists populated in sifAllServiceRequests.
countClosedSRs()	AccountServiceDetail	Integer	Static	Public	Array populated in sifAllServiceRequests.

About the Account Summary Smart Report

The report describes the relevant details pertaining to the account. It contains information about the account's historical and future revenue, satisfaction, organizational hierarchy, and service requests.

About the Order of Merit Graphic in the Account Summary Smart Report

The Order Of Merit graphic is determined in the Finish method of the ssOrderOfMeritHeader1 frame according to the following logic:

- If the Customer Satisfaction thermometer is above the target line, and the Pipeline thermometer is above the target line, then the arrow direction is up.
- If the Past Revenue thermometer is above the target line, and the Pipeline thermometer is below the target line, then the arrow direction is right.
- In all other cases, the arrow points down.

About the Past Revenue Thermometer in the Account Summary Smart Report

Table 27 shows the variables for use with the Past Revenue Thermometer.

Table 27. Variables for Past Revenue Thermometer

Thermometer Variable	Report-Level Variable	Calculation Point	Comment
MaximumValue	PastRevenueHigh	qryPastRevenueLostOfValues	Collected once per report.
MinimumValue	PastRevenueLow	qryPastRevenueLostOfValues	Collected once per report.
TriggerDataValue	PastRevenueAverage	qryPastRevenueLostOfValues	Collected once per report.
DataValue	PastRevenue	qryPastRevenue	Sum of the revenues for all opportunities associated with this account with 100% probability.

About the Pipeline Thermometer in the Account Summary Smart Report

Table 28 shows the variables for use with the Pipeline Thermometer.

Table 28. Variables for Pipeline Thermometer

Thermometer Variable	Report-Level Variable	Calculation Point	Comment
MaximumValue	PipelineHigh	qryPiplineListOfValues	Collected once per report.
MinimumValue	PipelineLow	qryPiplineListOfValues	Collected once per report.
TriggerDataValue	PipelineAverage	qryPiplineListOfValues	Collected once per report.
DataValue	PipelineRevenue	ftrPipelineThermometer	Sum of the revenues for all of the opportunities associated with this account.

About the Customer Satisfaction Thermometer in the Account Summary Smart Report

Table 29 shows the variables for use with the Customer Satisfaction Thermometer.

Table 29. Variables for Customer Satisfaction Thermometer

Thermometer Variable	Report-Level Variable	Calculation Point	Comment
MaximumValue	CustomerSatisfactionHigh	qryCustomerSatisfactionList_Of_Values	Collected once per report.
MinimumValue	CustomerSatisfactionLow	qryCustomerSatisfactionList_Of_Values	Collected once per report.
TriggerDataValue	CustomerSatisfactionTarget	qryCustomerSatisfactionList_Of_Values	Collected once per report.
DataValue	AverageScore	qryCustomerSatisfactionThermometer	The ratio of the total satisfaction score to the number of surveys.

About the Products Installed Graphic in the Account Summary Smart Report

The Products Installed graphic consists of controls generated dynamically in code overrides in the frmTimeLineAndHeader class. The OnRow method builds an array of unique install dates from the Install Date field of the Customer Product Business Component. The Finish method positions line and label controls along a time line based on the values in the array.

Two kinds of markers exist: year markers and install markers. Year markers are determined by iterating through the arrays, finding the oldest and newest install dates. The chart is scaled horizontally from these dates, and the individual install markers are placed proportionally.

About the Pipeline Analysis Smart Report

This report analyzes the current pipeline. Historical information is used to determine the revenue that is expected to be generated over the next four quarters and how that revenue compares to the quota for that period.

About the Order of Merit Graphic in the Pipeline Analysis Smart Report

The Order of Merit graphic is determined in the Finish method of the ssOrderOMeritHeader1 frame according to the following logic:

- If the Revenue versus All Current Quotas thermometer is above the target line, then the arrow direction is up.
- If the thermometer is between the target line and the acceptable revenue line, then the arrow direction is right.

- If the cementer is below the acceptable revenue line, then the arrow direction is down.

About the Revenue Versus All Current Quotas Thermometer in the Pipeline Analysis Smart Report

This thermometer has added functionality compared to standard thermometers. A dashed line is added at a point below the trigger to indicate a revenue level that is below the target but still acceptable. The position of the dashed line is based on the value of the AcceptableRange variable. Table 30 shows the variables for use with this thermometer.

Table 30. Variables for Revenue Versus All Current Quotas Thermometer

Thermometer Variable	Report-Level Variable	Calculation Point	Comment
MaximumValue	NA	local	TriggerDataValue * 1.5.
MinimumValue	NA	local	Always zero.
TriggerDataValue	totalQuota	sifQuota	Based on Prorated Target Amount field of Quota Objective Assignment Business Component.
DataValue	ExpectedRevenueNext4Q	frmDashboard	Based on the Probability Matrix derived from LOV.
AcceptableRange	targetQuotaRange	dsTargetQuotaRange	Based on value from LOV.

About the Quota Summary Smart Report

This report measures performance relative to quota for the current period and year-to-date. Closed revenue, as well as pipeline revenue, are tracked to determine whether the goal for the quarter can be met.

About the Order of Merit Graphic in the Quota Summary Smart Report

The Order of Merit graphic is determined in the Finish method of the ssOrderOfMeritHeader1 frame according to the following logic:

- If the Revenue versus All Current Quotas thermometer is above the target line, then the arrow direction is up.
- If the thermometer is between the target line and the acceptable revenue line, then the arrow direction is right.
- If the thermometer is below the acceptable revenue line, then the arrow direction is down.

About the Revenue Versus All Current Quotas Thermometer in the Quota Summary Smart Report

This thermometer has added functionality compared to standard thermometers. A dashed line is added at a point below the trigger to indicate a revenue level that is below the target but still acceptable. The position of the dashed line is based on the value of the AcceptableLevel variable. Table 31 shows the variables for use with this thermometer for the Quota Summary Report.

Table 31. Variables for Revenue Versus All Current Quotas Thermometer

Thermometer Variable	Report-Level Variable	Calculation Point	Comment
MaximumValue	NA	local	TriggerDataValue * 1.5.
MinimumValue	NA	local	Always zero.
TriggerDataValue	totalThermQuota	txtTotalQuota	Based on Quota field of the Opportunity Rollup.
DataValue	totalThermActualRevenue	txtActualRevenue	Based on ActualRevenue field of the Opportunity Rollup.
AcceptableLevel	acceptablePercentageOfQuota	frmDashboard	Based on value from LOV.
DataValue	ExpectedRevenueNext4Q	frmDashboard	Based on the Probability Matrix derived from LOV.
AcceptableRange	targetQuotaRange	dsTargetQuotaRange	Based on value from LOV.

About the Revenue By Direct Report Section in the Quota Summary Smart Report

This section contains special graphic components generated dynamically in the OnRow method of the frmDirectReportData component. The code sizes the Pipeline Revenue bar based on the sum of pipeline revenue and actual revenue. The Actual Revenue bar is sized according to the actual revenue value. The Quota Indicator line is superimposed in a position based on the quota value. The code also calculates the percent difference between the actual revenue and the quota, and determines whether or not the under-quota indicator will be displayed.

About the Service Request Performance Smart Report (Service Request Aging Analysis)

This report analyzes the aging of the currently open service requests. Three metrics are tracked that may explain performance: the number of open service requests, call volume, and average resolution time during the past months.

About Data Collection and Calculation in the Service Request Performance Smart Report

All service requests are collected and stored in a memory list in the rptAllServiceRequests report section. Later, in the rptServiceRequestDetail section, the service requests are pulled from the list, grouped by status and severity, and displayed in detail. As the service requests are collected, calculations are made and values stored in arrays for use in the TimeOpen, OpenServiceRequests, NewServiceRequests, and ResolutionTime sections.

About the rptAllServiceRequests Section in the Service Request Performance Smart Report

List storage, array storage, and calculations are accomplished in method overrides of the sifAllServiceRequests single input filter class. Table 32 shows the method overrides used for this class.

Table 32. Method Overrides for rptAllServiceRequests

Method	Scope	Comment
Fetch	sifAllServiceRequests	Stores a list of data rows for all Service Requests.
GetTimeOpenData	sifAllServiceRequests	Ignores closed service requests. Calculates time in days the service request was open. Categorizes the service request based on how many weeks it was open. Adds to the running count and total resolution time of service requests severity. Tracks high and low resolution times for severity.
GaetOpenSRData	sifAllServiceRequests	Calculates call volume by month and severity.
GetNewSRData	sifAllServiceRequests	Counts service requests that are less than six months old.
GetAverageResolutionTimeData	sifAllServiceRequests	Determines average resolution time by month and severity.

13 Parameterized Reports

This chapter describes creating parameterized reports in Siebel Tools and Actuate e.Report Designer Professional. The majority of steps for creating parameterized reports are the same as those for creating any other report. Therefore, both in Siebel Tools and Actuate, only the steps specific to parameterized reports are described in this chapter.

This chapter consists of the following topics:

About Parameterized Reports

Parameterized reports allow users to pass data into a report executable at runtime and customize the output of that report. The user may narrow the query, sort specification, or grouping by a field at the report's execution time. A parameterized report can produce different reports from the same report executable. The administrator defines the parameters and the attributes of a report during design time in Siebel Tools. For more information on report parameters, see the section on designing report parameters in the *Developing Advanced e.Reports* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

Siebel eBusiness Applications include several parameterized reports. Most of the parameterized reports are available in the Forecasting, Opportunity, and eAuction modules. When requesting a parameterized report, the Parameters window appears after selecting the report from the drop-down list. If the report is run using the Schedule Report option, the *Schedule a Report* window appears after the parameters on the Parameters window are specified.

NOTE: Some applications may not include parameterized reports.

Creating Parameterized Reports

Like all Siebel reports, parameterized report objects are first defined in Siebel Tools, and the corresponding ROL file is generated. As a prerequisite to creating a parameterized report, a specialized business component and an applet based on that business component are created. The business component contains fields that correspond to the report parameters. The applet contains the parameters and standard controls (such as Submit and Cancel). The reports created are included in the views that need to display them.

At the next step, the report design (ROD file) is created in Actuate e.Report Designer Professional referencing the parameters created in the ROL file. The report design file is then compiled with the Siebel libraries to create the report executable (ROX file). This report executable is deployed in the appropriate location (on the Siebel Reports Server or Siebel Mobile Web Client or both). When a parameterized report is run, the options on the parameter window allows the user to select or enter the values for the parameters included in the report design. The parameter variables and schedule variables (if any) are included in a report parameter file (ROV file) and submitted to the Reports Server. The report executable obtains these variables from the ROV file at report execution time. If the user runs the report interactively, the report is displayed in a DHTML viewer (in Siebel Web Client or Siebel Dedicated Web Client modes) or Siebel Report Viewer (in Siebel Mobile Web Client mode).

Siebel eBusiness Applications include out-of-the box parameterized reports. As part of customizing these reports, the reports administrator may add or delete the parameters in these reports. The administrator may also create custom parameterized reports. Parameterized reports are created and customized in Siebel Tools and Actuate e.Report Designer Professional as described in the following sections using the creation of the Revenue Analysis Summary report as an example.

Using Siebel Tools to Create Parameterized Reports

This section describes how to create and customize reports in Siebel Tools.

To create and customize parameterized reports in Siebel Tools

1 Create a business component object Revenue Report Parameters.

   Specify the properties of this business component as follows:

      Class and Name properties are required:

      Class = CSSBCVReportParameters.

      Name = Revenue Report Parameters. It is recommended that the business component be named to recognize that it forms the basis for the parameter applet for a specific report.

   Accept the default properties for all other properties of this business component, and any other business component created for a custom parameterized report.

2 Create the picklists to be used with the field object.

   In the Object Explorer select the Pick List object type and create picklists with the following properties:

      Picklist = Revenue Report By PickList; Revenue Report Show PickList; Revenue Report then PickList

      Business Component = PickList Generic for all picklists

3 Expand the business component in the Object Explorer and select Field. Create as many fields as the number of parameters in the report. The fields can be created as entry fields or picklists. For entry fields, the creation and association of picklists to business component fields can be skipped.

There are three parameters in the Revenue Analysis Detail report that allow filtering data at three levels. Therefore, create three fields with the following properties:

Name and picklist properties are required:

Name: By In Revenue; Show In Revenue; Then In Revenue.

PickList: Revenue Report By PickList; Revenue Report Show PickList; Revenue Report Then PickList.

4 Create an applet for the business component. The applet is the pop-up window that appears as the second or third screen when requesting a parameterized report.

Set the applet properties as follows:

Class: CSSSWEFrameReportParam. This class contains the logic for translating the selected parameters into the ROV file in Actuate.

HTML Popup Dimension: 200 x 200. This is the default size of the pop-up window, but this can be changed depending on the number of parameters.

Title: Revenue Parameter Applet. Set this property appropriately to represent the parameter applet for a custom parameterized report.

Business Component: Revenue Report Parameters. This is the name of the business component created for parameter applet in Step 1 on page 196.

5 Create an applet Web template to associate with the parameter applet.

a Highlighting the applet created in Step 4, expand the Applet object in the Object Explorer.

b Select Applet Web Template object and add two applet Web templates with the following required properties:

Type: Edit. This property specifies the mode in which this applet is rendered at run-time. Type should always be set to Edit since users need to select or enter values for parameters at run-time.

Web Template: Applet Popup Form. You may select any other Web template that has the desired layout for the parameter pop-up window.

6. Create the necessary controls for the Parameter pop-up window. These controls include labels, text, parameter fields, and standard controls. The properties for the controls for Revenue Parameter Applet are shown below:

   For standard controls:

   > Name: Submit Button; CancelButton. This property can be set to any meaningful name, but it is recommended that they indicate the function of the control.
   >
   > Method Invoked: Submit; CloseApplet. This property is required for standard controls and should be left blank for all other controls.
   >
   > HTML Type: MiniButton for both of the controls. This property should be set appropriately depending on the type of control. For example, set this value to FieldLabel for the label controls ByLabel, ReportLabel, ShowLabel, and ThenLabel.

   For parameter field controls:

   > Name: ByInRevenue; ShowInRevenue; ThenInRevenue. This property can be set to any meaningful name, but it is recommended that the name indicate the function of the control. In Control User Prop, make Name equal to SHOWNONE, where the value equals FALSE. This means that for the dropdown field shown on the UI, a blank entry is not allowed. This avoids the situation where a user clicks enter without selecting a valid value from the dropdown list.
   >
   > HTML Type: Field, for all three parameter field controls.
   >
   > Field: By In Revenue; Show In Revenue; Then In Revenue. Recall that these are the fields created in Step 2 on page 196. This property should be left blank for all other controls.

7. Create a report object to associate with the parameter applet. Set the Parameter Applet property of this report object to the name of the applet previously created.

8. Include the report object created in Step 7 in the views that should display this report.

9. Create the Actuate ROL file using Tools > Utilities > Generate Actuate Report.

The configuration in Tools is complete. The next steps involve creating the design in Actuate for the parameterized report.

Creating the Parameterized Reports in Actuate e.Report Designer Professional

Make sure that you have installed Actuate e.Report Designer Professional from the DVD-ROM. Since the report design process is similar to any other report, only the design specific to a parameterized report is described here. The description of creating parameterized report design in Actuate e.Report Designer Professional is described in the Designing report parameters chapter of Actuate's *Designing Advanced e.Reports* manual. The general process in designing a parameterized report includes two main steps:

- Creating parameters and referencing the parameters where they are used.
- Additionally, certain report methods to be modified using Actuate e.Report Designer Professional to pass the parameter values selected or entered by the Siebel applications user.

The first step is generic to all parameterized reports and described in Actuate’s *Designing Advanced e.Reports* manual. The second step is specific to Siebel parameterized reports and is described here for the Revenue Analysis Summary report.

When Revenue Analysis Summary report is run from a Revenues screen, the parameter screen appears after the report is selected from the Reports button’s drop-down list or after the schedule parameters are entered with the Schedule Report option. The user needs to select a value for each of the three parameters from the drop-down lists before clicking Finish.

The report is then generated and displays the first parameter field (which is the grouping field) in a cross tab format with the second parameter in aggregated date scale along columns, and the third parameter as rows. Therefore, this report is essentially a cross tab report (for more details, see Chapter 7, “Reports with Group Sections” in this book and the “Presenting data in crosstabs” section in the *Developing Advanced e.Reports* Actuate manual) with the displayed field selected by the user at run time. The second parameter defines the columns and the third parameter defines the rows.

To set report parameters

The report process parameters are set by the user in the client in the following order.

1. Store language dependent parameters fetched from the report ROV file (in the Start() method of root node ‘revd’) as global variables.

   ```
   Sub Start()

   ..........................

         ' get language values from user entered parameters

         ThenParameterV=ThenInRevenue

         ValueParameterV=ShowInRevenue

         ByParameterV=ByInRevenue

   ..............................................

   End Sub
   ```

2. Get the language independent codes for the parameter values selected by the user from the LOV table.

   ThenParameter = represents a group key.

   ValueParameter = represents a metric key.

   ByParameter = represents a date key.

   These values are necessary to support internal report logic that is language independent. The parameter retrieval is based on the ssSmart.rol library custom DataStream control ssList_Of_ValuesQuery (select ssSmart.rol from Library Organizer menu, double click on ssList_Of_ValuesQuery, and select Class tab).

   Modify the control methods Start() and Fetch() to set a search specification dynamically and store retrieved parameter language independent values.

```
Function Start() As Boolean
Dim DateSt as String
Dim ValueSt as String
Dim ThenSt as String

‘ create Search Spec date portion based on LOV or default ‘
‘ value
if revdet::ByParameterV <> "" Then
DateSt = "[Value] = '" & revdet::ByParameterV & "'"
else
DateSt = "[Name] = 'Month'"
end if

‘ create Search Spec Value portion based on LOV or default ‘
‘ value
if revdet::ValueParameterV <> "" Then
   ValueSt = "[Value] = '" & revdet::ValueParameterV &"'"
else
   ValueSt = "[Name] = 'Revenue'"
end if

‘ create Search Spec Then portion based on LOV or default ‘
‘ value
if revdet::ThenParameterV <> "" Then
   ThenSt = "[Value] = '" & revdet::ThenParameterV & "'"
else
   ThenSt = "[Name] = 'Account'"
end if

' dynamic search spec
```

```
SearchSpec="[Type] = 'REVN_FUNCTIONCAPTIONS' AND " & ValueSt
+ & " OR [Type] = 'REVN_SERIESCAPTIONS' AND " & ThenSt
+ & " OR [Type] = 'FCST_INTVL_PERD_TYPE' AND " & DateSt
Start = Super::Start( )
End Function

Function Fetch( ) As AcDataRow
   Dim aRow As ssList_Of_ValuesDataRow

   Set Fetch = Super::Fetch( )

   If Fetch Is Nothing Then
      Exit Function
   End If

   set aRow = fetch

   Select Case aRow.ssValue
      case revdet::ByParameterV
      revdet::ByParameter=aRow.ssName
      case revdet::ValueParameterV
      revdet::ValueParameter=aRow.ssName
      case revdet::ThenParameterV
      revdet::ThenParameter=aRow.ssName
   End Select
End Function
```

3 Use date parameter ByParameterV to set a search specification for ssPeriodQuery data stream from revper.rol. As a result Period business component returns only period record of specified type (day or month or quarter or year).

```
Sub Start( )
Super::Start( )
```

```
' set search spec to get only needed period type records

ssReport::ssSearchSpec = "[Period Type]='" &

revdet::ByParameterV & "' "

End Sub
```

4 Use ByParameter and ValueParameter parameter values to fill out revenue Line Item record dynamically. The purpose here is to create a list of revenue records that are independent of user parameters to make sure that general logic works correctly. Later, line item records are stored in a memory list for further aggregation.

```
Function CreateUListRow(Rec as acDataRow ) as uListRow

   Dim uRec as uListRow

   Dim aRec as ssRevenueDataRow

   dim CurrentSection as integer ' put into root


   ' calculate section key and UnitKeyStat

   CurrentSection=GetGroupKey()

   Set aRec = rec

   set uRec = New uListRow


   ' filter ByParameter

   Select Case LCase(revdet::ThenParameter)

      case "account"

         uRec.uThen=aRec.GetValue("ssAccount")

      case "campaign"

         uRec.uThen=aRec.GetValue("ssCampaign")

      case "opportunity"

         uRec.uThen=aRec.GetValue("ssOpportunity")

      case "project"

         uRec.uThen=aRec.GetValue("ssProject")

      case "partner"

         uRec.uThen=aRec.GetValue("ssPartner")

      case "product"
```

```
            uRec.uThen=aRec.GetValue("ssProduct")
         case "product line"
            uRec.uThen=aRec.GetValue("ssProduct_Line")
         case "description"
            uRec.uThen=aRec.GetValue("ssDescription")
         case "revenue type"
            uRec.uThen=aRec.GetValue("ssRevenue_Type")
         case "revenue class"
            uRec.uThen=aRec.GetValue("ssRevenue_Class")
         case "win probability"
            uRec.uThen=aRec.GetValue("ssWin_Probability")
         case "sales rep"
            uRec.uThen=aRec.GetValue("ssSales_Rep")
         case "contact last name"
            uRec.uThen=aRec.GetValue("ssContact_Last_Name")
         case "Quote"
            uRec.uThen=aRec.GetValue("ssQuote")
         Case Else
            if gErrorMsg = "" then gErrorMsg =
"'Then' parameter is invalid"
               Exit Function
   End Select

' filter ValueParameter
Select Case LCase(revdet::ValueParameter)
   case "revenue"
   uRec.uValue=toCur(aRec.GetValue("ssFunctional_Revenue_
   Formatted"))
   case "margin"
   uRec.uValue=toCur(aRec.GetValue("ssFunctional_Margin_
   Formatted"))
   case "cost"
   uRec.uValue=toCur(aRec.GetValue("ssFunctional_Cost_
   Formatted"))
   case "upside"
   uRec.uValue=toCur(aRec.GetValue("ssFunctional_Upside_
   Formatted"))
   case "downside"
   uRec.uValue=toCur(aRec.GetValue("ssFunctional_Downside
   _Formatted"))
   case "average price"
   uRec.uValue=toCur(aRec.GetValue("ssFunctional_Average_
   Price_Formatted"))
   case "quantity"
   uRec.uValue=toCur(aRec.GetValue("ssQuantity_Formatted"))

   Case Else ' default
   if gErrorMsg = "" then gErrorMsg = "'Show' parameter
   is invalid"
   Exit Function
   End Select
   ' Get values for the remaining fields
   uRec.uItem=aRec.GetValue("ssProduct")
   uRec.uCommit=aRec.GetValue("ssCommitted")
   uRec.uProb=aRec.GetValue("ssWin_Probability_Formatted"))
   uRec.uSalesRep=aRec.GetValue("ssSales_Rep")
   uRec.uRevCls=aRec.GetValue("ssRevenue_Class")
   uRec.uRevTp=aRec.GetValue("ssRevenue_Type")
   uRec.uCurrencyCode=aRec.GetValue("ssFunctional_Currency_Code")
   uRec.uDate=CDate(aRec.GetValue("ssDate_Formatted"))
   uRec.SectionKey = CurrentSection

   uRec.uDateUnit = UnitKeyStat ' GetUnitKey(uRec.uDate)


   ' create key for Item grouping based on displayed attributes

   ' Item+Commit+Prob

   uRec.ItemKey= Trim(uRec.uItem) & Trim(uRec.uCommit) &

   Trim(uRec.uProb)

   Set CreateUListRow = uRec

      End Function
```

Parts of the described approach to handling the user parameters can be of general use for other parameterized reports. They include Parameters retrieval, filtering Periods, and fetching values from the LOV table.

14 Developing Multilingual Reports

In the prior versions of Siebel eBusiness applications, a separate report design was needed for each language due to the hardcoding of user visible, language specific strings in the design. The format (locale) in which the fields were displayed was also hardcoded in the report design. As of release 7.5, users will be able to externalize the visible strings, locale information, and certain properties of data from the report design. At report generation time, the user interface strings of the desired language are obtained from an external file. The locale information is obtained from a locale map, and the report data is formatted.

The following topics are in this chapter:

Developing Multilingual Reports Overview

This section describes how extracting user-visible strings automatically from report design and Siebel report libraries can greatly simplify the localization process, reduce the time to create translations, and lower the localization cost. You need not be familiar with Actuate e.Report Designer Professional. The externalized strings of the reports can be translated and maintained separately without affecting the report design and the executable file.

New functions to export and import user interface strings have been implemented in sscustom.bas. When the report is first designed and run in the Design mode, all the control names, text strings, and certain user interface properties from the report are extracted to an external file. This external file is in plain text format and is referred to as an user interface file. You can then localize the text strings in this file. When the report is run in the Deployment mode (which is the mode in which report executables are deployed), the report will read the text strings from this user interface file.

Previously, to translate user-visible strings in a report, you would manually open each report design file and modify the properties of the label controls, text controls, and so on. With this user interface externalization feature, you will be able to translate the texts for label controls and the search alias for text controls in the user interface files that are created by the user interface externalization feature.

NOTE: When creating multilingual reports, the text file contains only translations for the label controls. When report data has changed, you do not need to copy anything back to or add anything to the text file. The only time you will need to add an entry in the text file is when you have added a new label control in the design, and you need to provide the language translation.

Designing Multilingual Reports

Designing a multilingual report is essentially the same as that of any standard report. The only additional step is to externalize the user interface elements for localization in the desired languages. The same report executable obtains the language-specific strings from the localized text files based on user's specification. The following instructions will describe the details of the additional step for localization of user interface strings.

To design multilingual reports

1. Design a report in the default language first; for example, ENU (English).

   If your installation uses a non-English version of Siebel eBusiness Applications, you do not have an \ENU folder. Instead, you have a folder in the appropriate language code for your installation, such as \DEU for Germany.

   See *Global Deployment Guide* for a list of three-letter International Standards Organization (ISO) language extensions.

   This part of the design is the same as already discussed in using Siebel Tools to create the ROL file and using Actuate e.Report Designer Professional to create the ROD and ROX files. For more information, see Chapter 6, "Creating a Simple List Report."

   a. Run the report in Design mode to externalize user interface strings and data properties to an external file.

   b. Set gExportUI to 1 in the beginning of the Start method of ssReport in sscustom.rol.

   c. Compile and run by setting ssLanguage to the default language; for example, ENU (English).

      The report executable and the user interface file are generated; for example, CUACCSVC.rox and CUACCSVC.txt.

      NOTE: This executable is for design time use only.

2. Localize the contents of CUACCSVC.txt into separate language specific files.

   a. Generate CUACCSVC.txt file specific to each language.

   b. Place each of the language specific TXT file in `\`*`Seibdev`*`\Tools\RPTSRC\`*`LANGUAGE`*`\CUACCSVC.txt`

      Where:

      LANGUAGE = the Siebel code for the language the report will use; for example, enu for U.S. English.

3. Run the report in Deployment mode to verify localization.

   a. Set gExportUI to 0 in the beginning of the Start method of ssReport in sscustom.rol.

   b. Recompile to create the report executable for the runtime deployment by setting ssLanguage to FRA (French), DEU (German), JPN (Japanese) and so forth as needed.

c. Verify that the language specific labels are displayed in the Actuate e.Report Designer Professional for each user specified language parameter.

NOTE: After testing a multilingual report, make sure that the report executable is created in the Deployment mode; for example, set gExportUI to 0 at the beginning of the Start Method in ssReport in sscustom.rol.

Deploying Multilingual Reports

With the report executable completed, you will need to package the reports for deployment to the Siebel eBusiness application.

To deploy multilingual reports

- Package the tested multilingual report.
  - Upload the CUACCSVC.ROX file (and any multilocale report) using Actuate Management Console into the Siebel Reports folder in the Report Encyclopedia.

    All language specific ROX files should be uploaded into Siebel Reports\LANGUAGE folder in the report encyclopedia.

    Where:

    LANGUAGE = the Siebel code for the language the report will use; for example, enu for U.S. English.
  - Save the CUACCSVC.txt files into `C:\sea7x\siebsrvr\REPORTS\`*`LANGUAGE`* directory for each language in the Actuate iServer host machine.

    The UNIX path will be `/export/home/actuate7/AcServer/lib/bin/ui/`*`language`* assuming Actuate iServer is installed in `/export/home/actuate7`.

    Where:

    LANGUAGE = the Siebel code for the language the report will use; for example, ENU for U.S. English.

Viewing Multilingual Reports

At report generation time, the enhanced user interface of the report generation wizard in Siebel eBusiness Applications now displays drop-down lists for the user to select the language and locale.

To view multilingual reports

1. Select the desired language and locale from the User Preferences screen Report Parameters view.
2. Select a multilingual report, which was developed previously, from a Siebel view.
3. Click the Reports button and select the report from the drop-down list.

4 Generate the report with the data and labels per specified language formatted to a specified locale.

   Locale information is read from a locale map.

   If no strings are found, labels and properties are defaulted to the original ROD file values for that report.

For more information, see "Requesting Reports in the Siebel Web Client" on page 37.

Exceptions for Multilingual Reports

Some exceptions exist when developing multilingual reports since some reports are not amenable to this single report ROX model. The following list consists of the exceptions for developing multilingual reports.

- Language specific address blocks are not adjusted automatically. For example, the ZIP code and telephone numbers fields are not adjusted to match the new language locale for the report.
- Fields based on locale are not automatically set to be included and or suppressed based on the locale selected for the report; for example, the JPN name alias field.
- Language specific search aliases are not automatically substituted.
- Reports that contain Charts and Graphs with labels will not reflect the locale of the selected locale for the report.

15 Report Business Service

This chapter describes the report business service methods. These methods can be used in scripts or workflow processes to automate reporting related business processes. For example, the administrator can define workflow processes to automate the business processes for generating a report with a specific query, or saving a report in the PDF format, or emailing a report to the customer.

This chapter consists of the following topics:

About the Report Business Service

Most users are familiar with report generation in the Siebel eBusiness Application views. In this mechanism, users may run a query in the view and then generate a report interactively or in the batch mode against that data. Subsequently, the report may be printed or shared with other Siebel users. Clearly this mechanism requires user interaction to accomplish reporting business needs. Using the report business service methods, customers can generate, share, and print reports automatically without user interaction. Since the report is automatically generated when certain business rules are satisfied, there is no way for the user to pass a query. Therefore, the view mode applied on the report executable is used for obtaining data.

Siebel Business Process Designer is an interactive software tool that lets you automate how your organization handles workflow processes.

For more information, please refer to *Siebel Business Process Designer Administration Guide* on the *Siebel Bookshelf*.

NOTE: Currently, the report business service methods are supported only on the runtime events. Therefore, a user must be logged on to trigger an event that will invoke these methods.

Knowledge of Siebel Tools, scripting, Siebel Business Process Designer and Siebel Reports Server is necessary to use the report business service methods. Having an understanding of running the business services is also necessary. Administrators can create as many workflow processes as needed to satisfy their business requirements and include the necessary report business service methods as steps (recall that workflow processes can include one or more business services as steps). The designers can test these workflow processes in the Business Process simulator. For more details on workflow processes and workflow policies, please refer to *Siebel Business Process Designer Administration Guide* on the *Siebel Bookshelf*.

NOTE: The report business service methods are only executed in the Siebel Web Client.

About Report Business Service Methods

The Report Business Service consists of methods that can be used in workflow processes or scripts. The following methods can be called in a workflow to automate report generation on the Reports Server.

NOTE: The report business service methods always operate on the Reports Server. Therefore, the administrator should make sure that the Reports Server is installed, configured, and is up and running.

Table 33 lists the available business methods.

Table 33. Report Business Service Methods Available for Reports

Name of Business Method	Purpose
DelOne	Deletes the specified user from the reports server encyclopedia and also removes the corresponding user folder. See "DelOne" on page 213 for more information.
DownloadReport	Downloads a report in PDF format from the reports server encyclopedia to the Siebel Server machine. See "DownloadReport" on page 213 for more information.
ExecuteReport	Runs a report on the Reports Server. See "ExecuteReport" on page 214 for more information.
GrantRolesAccess2Report	Grants access of report to roles (view only). See "GrantRolesAccess2Report" on page 215 for more information.
GrantUserAccess2Report	Grants access of report to other users (view only). See "GrantUserAccess2Report" on page 216 for more information.
PrintReport	Prints a report to a valid printer. See "PrintReport" on page 216 for more information.
RunAndEmailReport	Generates a report and sends it as an email attachment to one or more users. See "RunAndEmailReport" on page 216 for more information.
ScheduleReport	Schedules a report on the Reports Server. See "ScheduleReport" on page 218 for more information.
SyncOne	Synchronizes a single user on the Reports Server. See "SyncOne" on page 220 for more information.

About the Report Business Service Parameters

The following tables describe the parameters for the Report Business Service methods.

DelOne

Table 34 contains the input parameters for the DelOne business method.

Table 34. DelOne Input Parameters

Input	Description	Examples	Optional	Default
UserId	User to be deleted from the reports server encyclopedia.	SADMIN	No	

DownloadReport

Table 35 contains the input parameters for the DownloadReport business method.

Table 35. DownloadReport Input Parameters

Input	Description	Examples	Optional	Default
ReportRoiName	An existing ROI filename in the Actuate Encyclopedia.	`ReportName.roi`	No	Assume personal folder, for example `/myfolder/ReportName.roi`
Version	Version number of ROI.		Yes	0 (latest ROI is used).
TargetFileName	Name of output file in PDF.	`ReportName.pdf`	Yes	Default to `ReportName.pdf`
TargetDirectory	Directory on the Siebel Server machine where the report will be downloaded. This must be an existing directory.	`C:\reports`	No	

Table 36 contains the output parameter for the DownloadReport business method.

Table 36. DownloadReport Output Parameter

Output	Description	Example
OutputFileName	Full path to downloaded report	`C:\reports\ReportName.pdf`

ExecuteReport

Table 37 contains the input parameters for the ExecuteReport business method.

Table 37. ExecuteReport Parameters

Input	Description	Examples	Optional	Default
reportDefName	The name of the report object definition.	Employee Achievement Report	No	None
activeBOName	Name of Business Object.	Incentive Compensation Employee Position	No	None
reportName	Access Base DB Name.	`EMPACH`	Yes	Use information from report object definition.
OutputName	Full path/filename of the output ROI file.	`/SADMIN/OPLIST.roi`	Yes	Default folder to `/mylogin/reportName.roi`.
viewMode	View mode of report.	`0, 1, 2, 3,` and so on.	Yes	0
ssSearchSpec	Search specification of the report.		Yes	Use search specification on report object definition.
activeRowId	Active record row ID.		Yes	None
Locale	Locale	enu	Yes	Object Manager's language setting.
Language	Language	enu	Yes	Object Manager's locale setting.
ActuateServerHost	Actuate server host machine name.	bptu60s018	Yes	Use Object Manager's ActuateReportServerHost.
SSLoginName	Siebel Login.		Yes	User's login name, whose action triggers this business service.
SSPassword	Siebel Password.		Yes	User's password, whose action triggers this business service.

Table 37. ExecuteReport Parameters

Input	Description	Examples	Optional	Default
AcLoginName	Actuate login name.		Yes	Same as Siebel.
ActuatePassword	Actuate password.		Yes	Query from the business component RS Employee.
Print	A Yes or No flag for report printing.		Yes	
Printer	A valid printer name for Actuate Report Server.		Yes	
NumberOfCopies	Number of copies to be printed.		Yes	1
Concerning PollRequest Loop				
InitSleep	Initial sleep time (in seconds).	20	Yes	10
Intvl	Polling Interval (in seconds).	10	Yes	30
MaxWait	Time Out.	100000	Yes	1000000

GrantRolesAccess2Report

Table 38 contains the input parameters for the GrantRolesAccess2Report business method.

NOTE: The roles used for the RoleList input parameter needs to be defined in the Actuate Management Console by the administrator prior to being used for sharing reports.

Table 38. GrantRolesAccess2Report Parameters

Input	Description	Examples	Optional	Default
ReportRoiName	An existing ROI filename in the Actuate Report Encyclopedia.	`/SADMIN/OPLIST.roi`	No	None
RoleList	Comma-delimited list of roles.	`Director, President, Manager`	No	None
Version	Version number of ROI file.		Yes	0 (latest ROI file is used).

GrantUserAccess2Report

Table 39 contains the input parameters for the GrantUserAccess2Report business method.

Table 39. GrantUserAccess2Report Parameters

Input	Description	Examples	Optional	Default
ReportRoiName	An existing ROI filename in the Actuate Report Encyclopedia.	`/SADMIN/ OPLIST.roi`	No	None
UserList	Comma-delimited list of users.	`AFOSTER, KCHAN, RGAMPALA`	No	None
Version	Version number of ROI file.		Yes	0 (latest ROI file is used).

PrintReport

Table 40 contains the input parameters for the PrintReport business method.

Table 40. PrintReport Parameters

Input	Description	Examples	Optional	Default
NumberOfCopies	Number of copies to be printed.		Yes	1
OutputName	An existing ROI filename in the Actuate Report Encyclopedia.	`/SADMIN/ OPLIST.roi`	No	None
Printer	A valid printer name for the Actuate iServer.		No	User's default printer or Actuate Server default printer.
Version	Version number of ROI file.		Yes	0 (latest ROI file is used).

RunAndEmailReport

Emailing a report as an attachment to another user is available in the method RunAndEmailReport(). The attached report will be sent as an ROI file. For viewing the ROI file, download the Actuate Viewer from the Actuate web site (http://www.actuate.com/resourcecenter/index.asp) if you are using the Siebel Web Client. The Dedicated Web Client includes the Siebel Report Viewer that will allow viewing of the ROI file.

However, the following limitations do apply:

- You can only use this method for generating and then emailing that specific report. In other words, you will not be able to send a report attachment for reports that have already been generated.
- Related user preferences must be set manually within the Actuate Management Console. The report administrator has to specify how each user should handle report completion notification (see "Receiving Email Notifications and Adding Report Attachments" following this section).
- The email message (besides the particular report) can not be customized. The format is as provided in the Siebel application.

To configure Actuate to send email (using RunAndEMailReport), run `c:\Actuate7\Server\bin\mailinst.exe` and setup email profile.

NOTE: To enable email notification from the Actuate iServer, Microsoft Exchange Server must be used as the email Server. Email notification will not work if a standard SMTP email server is used. For more information about enabling email notification, see the *Administering Actuate iServer System* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

Receiving Email Notifications and Adding Report Attachments

For users to receive email notifications for both successful and failed requests and add reports as attachments (for successful cases), the Actuate administrator needs to make certain changes in the Actuate Management Console.

To receive email notifications and add report attachments

1. From the Start menu, open the Management Console and log in as Actuate administrator.
2. Click the Users icon on the left hand side.
   a. Choose a user and move the mouse pointer over the down arrow icon next to it.
   b. Select Properties from the drop-down list. Click the Jobs tab.
3. In the section For jobs that succeed, check both Send e-mail notification and Attach document options.
4. In the section For jobs that fail, check Send e-mail notification option.
5. Make sure that the Create completion notice option in both sections are checked (this is the default). Click OK.

Table 41 contains the input parameters for the RunAndEmailReport business method.

Table 41. RunAndEmailReport Parameters

Input	Description	Examples	Optional	Default
Include ALL relevant parameters in ExecuteReport( ) to specify the report to be run				
UserList	Comma-delimited list of users to receive email notices.	AFOSTER, KCHAN, RGAMPALA	No	None
ActuateServerHost	Actuate server host machine name.	bptu60s018	Yes	Use Object Manager's ActuateReportServerHost.
SSLoginName	Siebel Login.		Yes	User's login name, whose action triggers this business service.
SSPassword	Siebel Password.		Yes	User's password, whose action triggers this business service.
AcLoginName	Actuate login name.		Yes	Same as Siebel.
ActuatePassword	Actuate password.		Yes	Query from the business component RS Employee.

ScheduleReport

Table 42 contains the input parameters for the ScheduleReport business method.

Table 42. ScheduleReport Parameters

Input	Description	Examples	Optional	Default
reportDefName	The name of the report object definition.	Employee Achievement Report	No	None
activeBOName	Name of Business Object.	Incentive Compensation Employee Position	No	None

Table 42. ScheduleReport Parameters

Input	Description	Examples	Optional	Default
reportName	Access Base DB Name.	`EMPACH`	Yes	Use information from report object definition.
OutputName	Full path/filename of the output ROI file.	`/SADMIN/OPLIST.roi`	Yes	Default folder to `/mylogin/reportName.roi`.
viewMode	View mode of report.	0, 1, 2, 3, and so on	Yes	0
ssSearchSpec	Search specification of the report.		Yes	Use search specification on report object definition.
activeRowId	Active record row ID.		Yes	None
Locale	Locale	enu	Yes	Object Manager's language setting.
Language	Language	enu	Yes	Object Manager's locale setting.
ActuateServerHost	Actuate server host machine name.	bptu60s018	Yes	Use Object Manager's ActuateReportServerHost.
SSLoginName	Siebel Login.	enu	Yes	User's login name, whose action triggers this business service.
SSPassword	Siebel Password.	enu&&	Yes	User's password, whose action triggers this business service.
AcLoginName	Actuate login name.		Yes	Same as Siebel
ActuatePassword	Actuate password.		Yes	Query from the business component RS Employee.
StartDate	Starting date for report generation schedule.	1/1/2002	No	None
StartTime	Starting time for report generation schedule.	12:00:00	No	None

Table 42. ScheduleReport Parameters

Input	Description	Examples	Optional	Default
Repeat	A Y/N flag for recurring report generation.		No	None
Frequency	Frequency of report generation, for Repeat = Y.	Daily, Weekly, Monthly, Quarterly, and so on	No	None
UntilDate	Stop date for report generation schedule.	12/1/2004	No	None
NumberOfCopies	Number of copies to be printed.		Yes	1
Print	A Y/N flag for report printing.		No	None
Printer	A valid printer name for Actuate Report Server.		Yes	None

SyncOne

Table 43 contains the input parameters for the SyncOne business method.

Table 43. SyncOne Parameters

Input	Description	Examples	Optional	Default
AcAdminLogin	Actuate Administrator Login.	administrator	No	None
AcAdminPassword	Actuate Administrator Password.	““	No	None
UserID	User to be synchronized.	SADMIN	No	None
SendNoticeForSuccess	Indicates if notices should be generated for successful jobs.		Yes	FALSE
SendNoticeForFailure	Indicates if notices should be generated for failed jobs.		Yes	FALSE
SuccessNoticeExpiration	Expiration time in minutes for success notices.		Yes	0
FailureNoticeExpiration	Expiration time in minutes for failure notices.		Yes	0

Following is an example script for the SyncOne business method. When this script is run, you are returned feedback in the variable *sText*.

In the script below, SyncOne is executed on users VSILVER and DRA three times while storing the user feedback messages in the variable *msg*. You can consult *msg* on the status of the SyncOne request.

In this example, the script reflects three possible outcomes:

- The user is created.
- The user's password is reset.
- The script fails.

These outcomes are indicated in *msg*.

```
.....

var svc = TheApplication().GetService("Report Business Service");

var Inputs = TheApplication().NewPropertySet();

var Outputs = TheApplication().NewPropertySet();

var msg = "";

for(var x = 1; x < 3; x++)  {

   with (Inputs) {

      SetProperty ("AcAdminLogin", "administrator");

      SetProperty ("AcAdminPassword", "");

      SetProperty ("UserID", "VSILVER,DRA");

   }

   try {

      svc.InvokeMethod ("SyncOne", Inputs, ExtOutputs);

   }

   catch (e)

   {

      var sText = e.errText;

      var nCode = e.errCode;

      msg += sText;

   }

}
```

.....

For information on enabling these business methods, see information on object interfaces and scripting on *Using Siebel Tools* and in *Siebel Business Process Designer Administration Guide* on the *Siebel Bookshelf*.

Example: Invoking a Report Business Service Method Using Scripting

Workflow processes can be invoked programatically from a script using Siebel VB or Siebel eScript. By using scripts, workflow processes can be invoked from anywhere in the Siebel application or from external programs.

The following is a Siebel eScript code sample for scheduling the Employee Achievement Report to run at a specified time, one time only, using the ScheduleReport method.

```
var svc = TheApplication().GetService("Report Business Service");

var Inputs = TheApplication().NewPropertySet();

var Outputs = TheApplication().NewPropertySet();

var paramArgs  = TheApplication().NewPropertySet();


 with (Inputs) {

   SetProperty ("reportDefName", "Employee Achievement Report");

   SetProperty ("activeBOName", "Incentive Compensation Employee Position");

   SetProperty ("viewMode", "3");

   SetProperty ("StartDate", "05/01/02");

   SetProperty ("StartTime", "10:00:00 AM");

   SetProperty ("Repeat", "N");

   SetProperty ("Frequency", "");

   SetProperty ("UntilDate", "");

   SetProperty ("Print", "");

   SetProperty ("Printer", "");

}

Inputs.AddChild (paramArgs);

svc.InvokeMethod ("ScheduleReport", Inputs, Outputs);
```

A Library Reference

This appendix presents reference information about the library components in the sscustom and sssiebel libraries.

This appendix consists of the following topics:

- "About the sscustom Library" on page 223
- "About the sssiebel Library" on page 231

About the sscustom Library

Components in the sssiebel library are subclassed in the sscustom library. The components in the sscustom library may be further subclassed within sscustom.

All the components in the sssiebel library have a prefix of base; the corresponding descendent in the sscustom library start with ss. All additional descendents of ssTxt in the sscustom library are start with ssTxt.

This component relationship also applies to label controls, date controls, and the frame controls. The composite objects, ssReport and ssPageList, use slightly different naming conventions. The superclass for ssPage (landscape) is basePage, and ssPage is the superclass for ssPagePortrait. You must take care whenever you change the composite objects.

Text Controls in the sscustom Library

The following components are for displaying text:

- **ssTxt.** Basic text control
- **ssTxtB.** Bold text control
- **ssTxtBI.** Bold italic text control
- **ssTxtBIueBack.** Bold text control (white text on a blue background)
- **ssTxtChkBox.** Check box control
- **ssTxtP.** Percentage text control
- **ssTxtS.** Small text control
- **ssTxtSB.** Small bold text control
- **ssTxtSBI.** Small bold italic text control
- **ssTxtSectionHead.** Section head text control
- **ssTxtSectionHeadM.** Maroon section head text control

- **ssTxtSNoRepeat.** Small text control used in calendar reports
- **space.** Space control (aligns multiple text sections)

Actuate uses OLE automation to acquire data from the Siebel Application Server. In the first implementation of this architecture, all the data that is transferred is in the string data type format, including numeric and date information. The baseTxt control is the most appropriate method available for displaying string data.

Actuate can implicitly convert string data to other control types; for example, String data will be converted to the date data type if the control being used is a date control. Unfortunately, the implicit conversion methods for date and numeric data are not appropriate under some conditions, particularly when you are providing multilanguage support. See Figure 33.

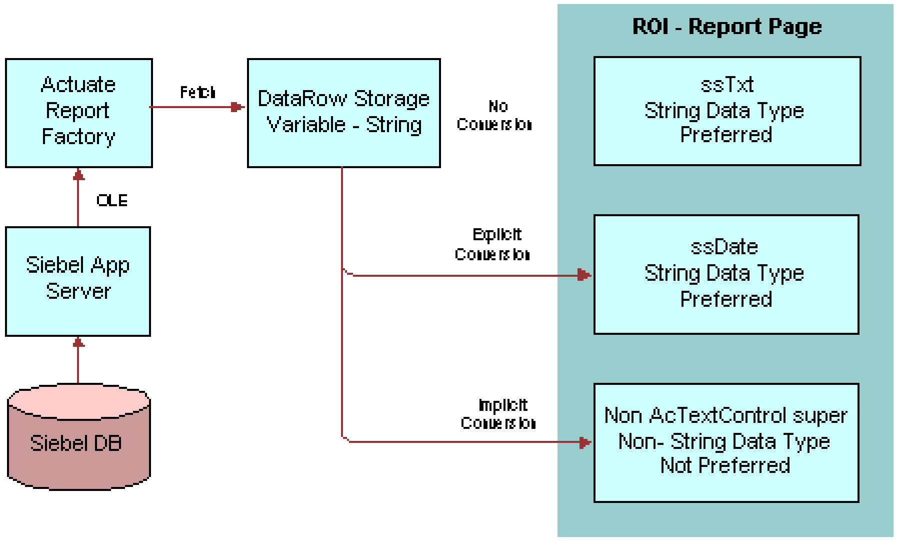

ROI - Report Page
Actuate Report Factory
Fetch
DataRow Storage Variable - String
No Conversion
ssTxt String Data Type Preferred
OLE
Siebel App Server
Explicit Conversion
ssDate String Data Type Preferred
Siebel DB
Implicit Conversion
Non AcTextControl super Non- String Data Type Not Preferred

Figure 33. Actuate Control Creation Process with Siebel Applications

Figure 33 shows the steps that data goes through before it populates an Actuate control. All data that is brought through the OLE interface is placed in DataRow Variables as string data. Then Actuate, in the course of instantiating controls to the ROI page, takes the data from the row and populates the various controls.

As long as the information to be displayed is string data, no conversion takes place. This provides the best support for multiple languages and is the preferred method.

Date information and revenue information are handled differently. Siebel Systems has developed methods that are appropriate for working with both date and revenue information.

How to Display Revenue Information in Siebel Reports

When you are creating datastreams with Siebel Tools, each field is mapped to a data row variable, of string data type. If the field is defined in the Siebel application as being a currency, then Siebel Tools will map two string variables into the data row:

- ssFieldName
- ssFieldName_Formatted

The standard revenue contains the value of the field, contained in a string (for example, “1243” or “1000000”). The formatted revenue will contain the value of the field with the appropriate formatting information for currency for that record (for example, “$1,243” or “£1.000.000”). As long as the data does not need to be used in a calculation, the formatted data field is the preferred display method. Standard revenue field (unformatted) is not used at all in reports due to its localization inability.

The following explains how Siebel reports handle calculated currency.

There are three steps involved in the calculation and display of currencies:

1. Convert the formatted currency string fetched from the Siebel application to a currency data type value that can be used in arithmetic operations, using the ToCur( ) wrapper.
2. Perform any calculations.
3. Display the currency data using txtCurrency as the superclass for the layout field. This changes the calculated currency value back to a formatted string for display purposes. The details are given below.

   Formatting currency string to number for calculation:

   > Problem: Actuate conversion functions such as CCur( ) or CDbl( ) do not correctly interpret a formatted currency string if any text is attached to the value.
   >
   > For example, CCur(DM 55,77) yields 0.
   >
   > Solution: To assure proper conversion from string to number, use ToCur( ) defined in sssiebel.bas. This function removes any nonnumeric content from the string and then applies CCur( ) to the input string.
   >
   > For example, ToCur(DM 55,77) yields 55,77

   Formatting number to currency for display:

   > Problem: Actuate cannot format currency values for a specific currency code (Actuate can do it for the system currency only).
   >
   > Solution: Use FormatSpecificCurrency( ), defined in sscustom.bas to make sure there is proper currency value formatting for a specified currency code.
   >
   > Example: cur = FormatSpecificCurrency(55.77, USD)

One more issue to handle for currencies is multicurrency calculations.

Multicurrency processing is a long-existing issue for reports because of their inability to do conversion between currencies. A solution is to have two values (item and system currency) for any currency type field (revenue, cost, . . .). This way, we have a unified approach to the aggregation of different currency values. System currencies display line items in their item currency formats, and aggregate values are displayed in the system currency format. To support this idea, the business component that supplies a report with data must have each currency field in two flavors: Function-Currency-Value (system-based) and Currency-Value (item currency).

If the text control represents the sum of multiple rows of data in the AFTER section, use the numeric revenue amount. The ValueExp property can be formulated as follows:

```
FormatSpecificCurrency(Sum(CurrencyValue),”USD”).
```

Actuate cannot sum formatted data, so the expression shown previously will be formatted by the FormatSpecificCurrency( ) function described in the sscustom.bas.

For multicurrency calculations, include the Functional Revenue (the revenue value in the functional currency) and the Functional Currency Code (for example, USD) in your report and subreport fields when you define the ROL file in Siebel Tools. For an example of multicurrency calculations, see the Pipeline Report By Rep (PIPEREP) standard report.

How to Display Date Information in Siebel Reports

NOTE: The information described in this section concerning dates is for legacy Siebel reports. In version 7 reports, all newly created date fields must use Formatted date fields from Application and ssDateTimeControl from sscustom. This new Actuate control has verity of format options to pick including Short-Date, Long-Date and Medium-Date (same for time).

Date information is more complicated than revenue data. As with the currency information, each Report Object field is mapped to two variables in the data row:

- **ssDateFieldName.** The standard field is in the format of Siebel’s internal date representation and is always in the format MM/DD/YY HH:MM:SS.
- **ssDateFieldName_Formatted.** The formatted field comes in the format specified for the business component that is used to get the data, and it uses the locale settings of the machine. If the screen displays dates using the system short date format, the DataRow variable will be populated with 31/12/01, 12/31/01, or 01/12/31, depending on the locale setting of the machine.

  The key factor is that the string does not need any additional formatting and can be displayed using the ssTxt control. Attempting to use the ssDate control with Date_Formatted information will cause a fatal error within Actuate e.Report Designer Professional or the Siebel application.

  NOTE: Use ssTxt when using the formatted date.

To display the full month name, date, and year (November 24, 2001), use ssDate.

The ssDate control forces an explicit conversion of the string date expression using known settings. The string is then converted back into string format and displayed using the format from the Format property. If the string is an empty string, the control is detached from the frame so that no data is shown.

Five predefined date formats are used within the standard reports. ssDate, ssDateLongDate, and ssDateShortDate all use date settings from the regional locale settings on the machine. ssDateHNNAP and ssDateMMMMDDYYYY use hard-coded format strings to display the information in a specialized format. The date library components are the following:

- **ssDate.** Short date and standard time
- **ssDateHNNAP.** Time with AM or PM, as in 12:55 AM
- **ssDateLongDate.** Long date (regional setting)
- **ssDateMMMMDDYYYY.** Date with text month, as in November 24, 2001
- **ssDateShortDate.** Short date (regional setting)

You can modify the date formats in the sscustom library by changing the Format property of the date control. New date formats can be added by subclassing from ssDate. To change the Short Date or Long Date displays, change the regional settings on the machine.

About the CanGrow Property

The CanGrow property is available to all ssTxt and ssDate controls. Presently, this availability is only in the Windows environment. In the UNIX environment, the CanGrow property is not available with the Reports Server.

About Check Box Text Control

The ssTxtChkBox control displays Boolean information. A check mark is displayed whenever TrueCondition is TRUE. TrueCondition evaluates the DataValue of the control with the string expression in the TrueCondition Property; if they match, the condition is TRUE and the check mark is displayed.

You can adjust TrueCondition by changing the value of the TrueCondition property. The default property value is Y.

The ssTxtChkBox control uses the Monotype Sorts font to display check marks. If the Monotype Sorts font is unavailable on the local machine, a 3 is displayed for TRUE conditions and “” for FALSE conditions. The code that controls this can be adjusted in the OnRow method of the control.

About the Percentage Text Control

The percentage text control is used to display string numeric data as a percentage. Table 44 shows the various rules for displaying the percentage strings.

Table 44. Percentage Data Conversion

Input Value	Output Value
0	0
3.0	3%
3.3	3.3%

The code used to format the data is in the OnRow method and can be modified to change the behavior for the control and its children.

About the Label Controls

The following label components are available in the sscustom library:

- **ssLblB.** Bold label control
- **ssLblBI.** Bold italic label control
- **ssLblHead.** Header label control
- **ssLblHeadBlack.** Header label control
- **ssLblQuotation.** Quotation label control
- **ssLblSB.** Small bold label control
- **ssLblSBI.** Small bold italic label control
- **ssLblSectionHead.** Section heading label control

The label control displays fixed text strings that you set at design time. To change the text of an individual control, you modify the text property in the component editor window or you modify the control directly by single-clicking the control twice.

The advantage of using multiple types of label controls is that it increases maintainability within the entire report library. For example, to change a font in all the text controls, you simply modify the font.FaceName property in the ssTxt control. To change all the row headings within the reports, you change the desired property in the ssLblHead control.

The controls in the sscustom library have intentionally been made generic and simple. The small bold label control (ssLblB) can be any font size, although the standard reports use a small font size of 8 points.

About the Frame Controls

The following frame components are available:

- **ssFrmRecordSeparator.** Landscape frame that contains a line
- **ssFrmBlueBack.** Landscape frame with blue background
- **ssFrmBlueBackP.** Portrait frame with blue background
- **ssFrmGrayBack.** Landscape frame with gray background
- **ssFrmGrayBackP.** Portrait frame with gray background

All the ssFrm controls inherit from the baseFrm class. Any text control that is using the CanGrow function (CanGrow = TRUE) requires that its container frame be of type baseFrm or a type derived from baseFrm. If the container frame is not of type baseFrm, a run-time error will occur.

Use a portrait mode frame if the report is designed for portrait orientation.

About the PageList and Child Components

The page controls are perhaps the most complicated feature in the Siebel report libraries. By default, Siebel reports use the ssPageList control. The pageList control is a container control that holds the list of all the pages. As each page is instantiated, it is added to the list as a persistent object.

The pageList component can add pages in landscape mode, portrait mode, or any custom page layout desired. Siebel reports ship with two basic page styles, the landscape style (ssPage) and the portrait style (ssPagePortrait). Landscape is the default page style.

To switch from landscape style to portrait style

1. Click the ssPage component, under the ssPageList component, and click Delete.

   This will drop the ssPage component and leave the slot blank.

2. Drag the ssPagePortrait component from the library and drop it onto the ssPageList component.

   The report will now use a portrait-style page. Make sure that the frames used in the report have portrait dimensions.

To create a different page layout, you subclass from the basePage component. To create a new page from scratch that does not use any of the existing controls, you subclass directly from the AcSimplePageList control.

All the portrait style controls are inherited from the landscape controls. To modify the existing page layout, change the ssPage layout, then determine whether changes to the ssPagePortrait component are required.

All other controls on the ssPage and ssPagePortrait components descend directly from the sssiebel library components. Inheritance of page control components is as follows:

- basePageList > ssPageList
- basePage > ssPage > ssPagePortrait

- baseFlow > ssFlow > ssFlow1 > ssFlowP
- baseReportHeaderBar > ssReportHeaderBar > ssTitleBarP
- basePrintBy > ssPrintBy > ssPrintByP
- baseDateDisplay > ssDateDisplay > ssDateDisplayP
- basePageNoDisplay > ssPageNoDisplay > ssPageNoDisplay
- baseRptCreateBy > ssRptCreateBy > ssRptCreateByP
- baseReportTitle > ssReportTitle > ssReportTitleP
- baseLblSiebel > ssLblSiebel > ssLblSiebelP

About Miscellaneous Controls

All of the following controls have been placed in the sscustom library to allow future enhancements. These controls are not currently used in the standard reports:

- **ssCur.** Currency control.
- **ssCurB.** Bold currency.
- **ssFloat.** Float control.
- **ssInt.** Integer control.
- **ssSubPage.** Subpage.
- **ssImageControl.** Image control with no changes, placed for future support. The Image control is used to display bitmap images on the screen.
- **ssSummaryGraph.** Graph control with enhanced Y-label support. The Graph control used in Siebel applications has been modified to allow increased control of the Y-labels.

About Line Controls

The line controls are straightforward. Some property-level changes give the controls varying behavior or appearance.

- **LineControlP.** Line control with portrait width
- **DblLine.** Double-line control

About Section Components

The conditional and parallel sections have special roles:

- **ssConditionalSection.** Conditional section. The conditional section control allows the selection of one of two frames based on a run-time condition. This function is displayed in the Quote Configuration report.

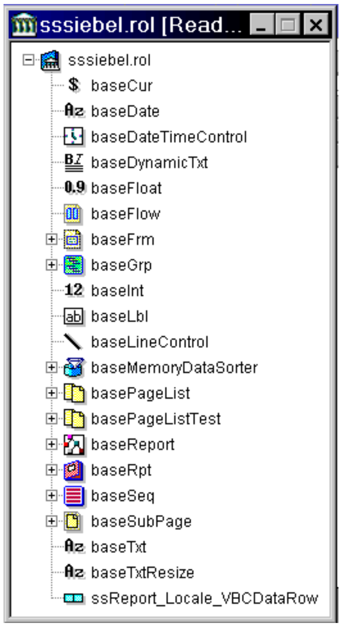
- **ssParallelSection.** Parallel section. In general, frames are placed on the page sequentially. Parallel sections allow multiple frames to be filled on a page at the same time (that is, in parallel). See the Actuate product documentation for further information on the use of parallel sections.

The group, sequential, and report section controls inherit all their properties and methods directly from their superclasses in the sssiebel library without modification.

- **ssGrp.** Group section
- **ssSeq.** Sequential section
- **ssRpt.** Report section

About the sssiebel Library

All the commonly used components in the Actuate Foundation Class (AFC) library have been extended into the sssiebel library, shown in Figure 34. This section provides a list of the components in the sssiebel library and explains how they have been modified.



sssiebel.rol [Read...

sssiebel.rol
- baseCur
- baseDate
- baseDateTimeControl
- baseDynamicTxt
- baseFloat
- baseFlow
- baseFrm
- baseGrp
- baseInt
- baseLbl
- baseLineControl
- baseMemoryDataSorter
- basePageList
- basePageListTest
- baseReport
- baseRpt
- baseSeq
- baseSubPage
- baseTxt
- baseTxtResize
- ssReport_Locale_VBCDataRow

Figure 34. The sssiebel Library

baseCur

Superclass	AcCurrencyControl	
Properties	Font.FaceName	Arial
	Font.Size	10
Methods	None	

Variables None

Notes Not used in existing standard reports but included in the library for completeness and possible future use. Components derived from baseTxt are used to display currency fields as described in “About the sscustom Library” on page 223.

baseDate

Superclass baseTxt

Properties None

Methods OnRow

Variables None

Notes Parses the standard date string in ‘MM/DD/YY HH:MM:SS’ into a serial date data type. It then formats the date as a string according to the current locale setting. If the string is blank, the width becomes zero.

DateDisplay

Superclass baseTxt

Properties

LabelPrefix	“Date “
ValueExp	Format(Now(), “Short Date”

Methods Finish

Variables

LabelPrefix	Property	String
LabelSuffix	Property	String

Notes Shows the report run date. The Short Date format is used as specified by the locale setting. baseDateDisplay concatenates the values in the LabelPrefix string and the labelSuffix string:

LabelPrefix & Format(Now(), “Short Date”) & LabelSuffix

baseFlow

Superclass AcTopDownFlow

Properties Position, Size

Methods AddHeader

Variables None

Notes The AddHeader method is modified to work with the CanGrow functionality.

baseFlow1

Superclass baseFlow

Properties None

Methods None

Variables None

Notes The flow represents the area on each page in which the information from the data rows is displayed. baseFlow1 has been subclassed for future flexibility.

baseFrm

Superclass AcFrame

Properties Size
AlternateColor
AlternateLines

Properties
Size
AlternateColor
AlternateLines

Methods
AdjustAssociatedControl
AdjustControlPositions
AdjustSize
BindToFlow
GrowFrame
Start

Variables	AlternateColor	Property	AcColor
	AlternateLines	Property	Integer
	MostRecentContainer	Public	AcReportComponent
	MostRecentFlow	Public	AcFlow
	OriginalSize	Public	AcSize
	RowNumber	Public	Integer
	SomeThingGrew	Public	Boolean

Notes baseFrm has been modified to support CanGrow functions; no user-level adjustments should be made to the CanGrow function.

baseFrm no longer supports alternate line colors as of the Siebel 7 release.

baseGrp

Superclass AcTopDownFlow

Properties Page.ShowHeaderOnFirst True

Methods
StartFlow
StartGroup

Variables	None	
Notes	baseGrp has been modified to make it compatible with the CanGrow functionality.	

baseInt

Superclass	AcIntegerControl	
Properties	Font.FaceName	Arial
	Font.Size	10
Methods	None	
Variables	None	
Notes	Not used in existing standard reports.	

baseLbl

Superclass	AcLabelControl	
Properties	Font.Size	10
Methods	None	
Variables	None	

baseLblSiebel

Superclass	AcImageControl	
Properties	FileName	“...\lib\ssLogo.bmp”
Methods	None	
Variables	None	
Notes	Displays the Siebel corporate logo. Change FileName to place a different corporate logo on the report.	

baseLineControlr

Superclass	AcLineControl	
Properties	Position	
	EndPosition	
Methods	Start	

Variables None

basePage

Superclass	AcPage	
Properties	Size.Height	8.27
Methods	None	
Variables	None	

Notes Determines the report’s page dimensions. Landscape mode is the default. Portrait mode is available in the sscustom Library.

The basePage uses the least common denominators between A4 and 8.5x11 so that the reports will display on a page of either size, as shown below:

	8.5 x 11	A4	Siebel Actuate Reports
Height	8.5	8.27	8.72
Width	11	11.69	11

If the reports will be printing on predominantly A4-size pages, you can modify the Position variable in the sscustom library to center the information on the page.

basePageList

Superclass	AcSimplePageList		
Properties	LabelMidString	“of “	
	LabelPrefix	“Page “	
	PageStyle	basePage	
	PageXofX	True	
	cstr_PageNoObject	ssPageNoDisplay	
Methods	AddFrameToFlow		
	Finish		
	SetPageXofX		
	Start		
Variables	cstr_PageNoObject	Property	String
	LabelMidString	Property	String
	LabelPrefix	Property	String
	LabelSuffix	Property	String
	PageXofX	Property	Boolean

Notes The key feature of the basePageList component is the PageXofX property. If PageXofX is TRUE, the component specified in cstr_PageNoObject displays the following string:

LabelPrefix & curPageNum & LabelMidString & TotalPages & LabelSuffix

The default display reads as Page X of Y, where X is the current page number and Y is the total number of pages.

The cstr_PageNoObject variable is the name of the component that holds the page number.

basePageNoDisplay

Superclass	baseTxt		
Properties	None		
Methods	None		
Variables	None		
Notes	basePageNoDisplay is a blank control that is filled in by the basePageList.PageXofX code.		

basePrintBy

Superclass	baseTxt		
Properties	LabelPrefix	“Printed By “	
Methods	Finish		
Variables	LabelPrefix	Property	String
	LabelSuffix	Property	String
Notes	basePrintBy uses a parameter passed by the Siebel client to show the user name of the person running the report. If the client fails to provide a user name for the report, the component displays the name of the user on the local operating system. As with the page number control, the basePrintBy control concatenates LabelPrefix, UserName, and LabelSuffix into one string, such as “LabelPrefix & UserName & LabelSuffix.”		

baseReport

Superclass	AcReport	
Properties	Content	baseRpt
	PageList	basePageList
Methods	Start	

Variable	pubReportTitleg	Public Variable	String
	ParamLocale	Parameter	String
	ssActiveRowId	Parameter	String
	ssBookmark	Parameter	String
	ssBusObjectName	Parameter	String
	ssDataLanguage	Parameter	String
	ssLanguage	Parameter	String
	ssLocale	Property	String
	ssOLEServer	Parameter	String
	ssPassword	Parameter	String
	ssPositionId	Parameter	String
	ssReportName	Parameter	String
	ssReportTitle	Parameter	String
	ssSearchSpec	Parameter	String
	ssSiebelSever	Parameter	String
	ssSortSpec	Parameter	String
	ssUserName	Parameter	String
	ssViewMode	Parameter	String

Notes The baseReport component is required for all Siebel reports. The Siebel client passes these parameters to the report when a report is initiated. The report will not run if parameters are not defined correctly.

The ssReportTitle property allows the user to set the report name, independent of the report object definition in the repository. The report title is displayed in the header of the report and can be modified in the ssReport object for language customizations or company-specific name changes.

The Start method moves the value in the ssReportTitle variable into the pubReportTitle variable so that it can be accessed from anywhere within the report.

baseReportHeader

Superclass baseLbl

Properties None

Methods None

Variables None

Notes baseReportHeader is the solid black bar that appears at the top of every report.

baseReportTitle

Superclass baseLbl

Properties None

Methods Finish

Variables	None		
Notes	The baseReportTitle control looks up the name of the report specified in the ssReportTitle property of the ssReport (top object) and displays that information in the header of the report.		

baseRpt

Superclass	AcReportSection		
Properties	Page.ShowHeaderOnFirst	True	
	Page.ShowFooterOnLast	True	
Methods	BuildOnePass StartFlow StartGroup		
Variables	FirstRowFetched	Public	Boolean
Notes	baseRpt has been modified to work with the CanGrow functionality in the text control.		

baseRptCreateBy

Superclass	baseLbl	
Properties	Text	“Report Created by Siebel Software”
Methods	None	
Variables	None	
Notes	baseRptCreateBy is a label string that can be used to add any text message that is common to every report.	

baseSeq

Superclass	AcSequentialSection	
Properties	TocAddComponent	True
	TocAddContents	True
Methods	None	
Variables	None	

baseSubPage

Superclass	AcSubpage
Properties	None
Methods	None
Variables	None

baseTxt

Superclass	AcTextControl	True	
Properties	CanGrow	True	
	CharacterWrap	True	
	Font.FaceName	Arial	
	Font.Size	10	
	TextPlacement.MultiLine	True	
	TextPlacement.WordWrap	TextWordWrap	
Method	BuildFromRow		
	GetFormattedText		
	GetText		
	GrowControl		
Variables	AssociatedControl	Property	String
	CanGrow	Property	Boolean
	CharacterWrap	Property	Boolean
	ControlAdjusted	Public	Boolean
	OriginalSize	Public	AcSize

Notes

By default, every Siebel text control can grow so that all the text in the control is displayed. You can change this function by modifying the CanGrow and CharacterWrap properties.

When a control grows, it adjusts its container (frame) to fit the new control size. In addition, it adjusts any controls that are underneath it to make room for the adjusted control.

You can enter the name of a label control in the AssociatedControl property to specify a persistent association between label control and text control.

B Method Reference

This appendix defines the DataStream function used when developing reports using Actuate. Also described are the three datastream methods used with Siebel standard reports. Sample code is included for each of the datastream methods. For more information about using datastream methods, see the *Developing Advanced e.Reports* manual in the Actuate folder of the *Siebel eBusiness Third-Party Bookshelf*.

This appendix consists of the following topics:

- "About the DataStream Function" on page 241
- "About the Essential DataStream Methods" on page 241

About the DataStream Function

In Actuate, the term *datastream* refers to a collection of components that deliver data to the report. Because Siebel data access is accomplished through the Siebel Object Manager, queries are executed outside the Actuate environment. The role of the datastream is to create the Siebel Object Manager interface and deliver the formatted data row to the report design. Siebel Tools defines the variables required to execute the Siebel Object Manager automation; their values are determined by the requirements of the report design. The datastream variables, with their types and functions, are listed in Table 45.

Table 45. Siebel Datastream Variables

Variable	Type	Function
SsAppServer	Integer	Created in the Start method of the datastream; set to the active Siebel application server.
ssBO	Integer	Created in the Start method; holds the value of the active business component.
SsBusCompName	Integer	Holds the value of the master business component for the active view in the Siebel client.

About the Essential DataStream Methods

Three essential datastream methods are used in Siebel standard reports:

- Start
- Fetch

- Delete

These are described in the sections that follow.

About the Start Method

The Start method has two prominent roles. It creates the Siebel Object Manager interface objects required to populate the data row, and it makes sure that the required fields are active. The Start method is called only on the master datastream. If a subreport uses the same business object, the subreport (child) datastream uses the interface created by the parent (master) datastream. Similarly, since fields are activated in the Start method, the master datastream activates fields for all its children.

NOTE: The code is written such that the user's screen does not change while the report is being generated, even if a subreport uses a different business component.

The following code sample is from the Start method in the ssAccountQuery datastream component in the Aclist (Account List) report.

```
Function Start () As Boolean


    Dim bCurRowOnly  As Boolean
    Dim bReExecute   As Boolean
    Dim bUpdateLinks As Boolean
    Dim bStatus      As Boolean
    Dim mainRowId    As String
    Dim busObjName   As String
    Dim SortSpecDyn  As String
    Dim searchSpec   As String
    Dim curViewMode  As Integer


    bCurRowOnly  = False
    baseReport::bCurRowOnly   = bCurRowOnly
    bReExecute   = False
    baseReport::bQueryExecuted = False
    bUpdateLinks = False
    errCode      = 0
```

```
   mainRowId    = ""
   busObjName   = ""

   bStatus = Super::Start ()
   If (bStatus = False) Then
      Start = False
      Exit Function
   End If

   ssAppServer = ssReport::ssiSiebelServer

   ssModelSetPositionId (ssAppServer, ssReport::ssPositionId)

   ssBO = ssModelGetBusObject (ssAppServer, ssReport::ssBusObjectName)
   ssReport::ssActiveBusObject = ssBO

   ' SearchSpec is recieved depending on where it is originated
   SearchSpec = ""
   If (ssReport::ssSearchSpec <> "") And (errCode = 0)  Then searchSpec = 
ssReport::ssSearchSpec

   ' SortSpec is recieved depending on where it is originated
   SortSpecDyn = "Name"
   If (ssReport::ssSortSpec <> "") And (errCode = 0)  Then SortSpecDyn = 
ssReport::ssSortSpec

   If (errCode = 0) Then ssAccount = ssBusObjGetBusComp (ssBO, "Account")
   If (errCode = 0) Then ssBusCompSuppressNotification  (ssAccount)
   If (errCode = 0) And (ssType = 2) Then ssBusCompDeactivateFields (ssAccount)
   If (errCode = 0) Then curViewMode = ssBusCompGetViewMode (ssAccount)
   ' If the main report BusComp has a search or sort spec,

   ' or uses a field that isn't already active, its query

   ' must be re-executed.

   If (ssType = 2) Then bReExecute = True

   If (Not bReExecute)  Then bReExecute = (searchSpec <> "") Or (SortSpecDyn <> "")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsExecuted", "") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Account Status") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "City") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Competitor") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Country") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Description") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Industry") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Location") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Main Fax Number") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Main Phone Number") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Name") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Parent Account Name") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Postal Code") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "State") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Street Address") = "N")

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Synonym") = "N")
```

```
   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "IsFieldActive", "Type") = "N")

   ' Last check if business component needs to be re-executed

   If (errCode = 0) And (Not bReExecute) Then bReExecute = (ssBusCompInvokeMethod
(ssAccount, "ReExecuteReport", "") = "Y")

   ' Cache the current row-id so it can be re-queried

   If (bCurRowOnly) Then mainRowId = ssReport::ssActiveRowId


   If (errCode = 0) And (ssType = 1) And (bReExecute) Then

      ' To re-execute a current-row-only report, the active

      ' BusComp need not be disturbed.  Another BusObj can

      ' be constructed just for this query.

      ' This is NOT applicable for server based report generation.

      If (bCurRowOnly) And (ssReport::ssBusObjectName = "") Then

         baseReport::bQueryExecuted = True

         If Not ( ssAccount = 0 ) Then ssBusCompAllowNotification (ssAccount)

         If (errCode = 0) Then busObjName = ssBusObjGetName (ssBO)

         If (errCode = 0) Then ssBO = ssModelGetBusObject (ssAppServer, busObjName)

         If (errCode = 0) Then ssAccount = ssBusObjGetBusComp (ssBO, "Account")

         If (errCode = 0) Then ssBusCompSetViewMode (ssAccount, curViewMode)

         If (errCode = 0) Then ssBusCompSuppressNotification  (ssAccount)

      End If

   End If


   ' Now actually setup the main report BusComp

   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Account Status")

   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "City")

   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Competitor")

   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Country")

   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Description")

   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Industry")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Location")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Main Fax Number")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Main Phone Number")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Name")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Parent Account Name")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Postal Code")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "State")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Street Address")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Synonym")
   If (errCode = 0) Then ssBusCompActivateField (ssAccount, "Type")


  If (errCode = 0) Then ssBusCompInvokeMethod (ssAccount, "SetIgnoreMaxCursorSize", "Y")

   If (errCode = 0) And (ssType = 2) And (bCurRowOnly = FALSE) Then
      If (baseReport::bSetForward) Then
         ssBusCompInvokeMethod(ssAccount, "GotoBookmarkT", ssReport::ssBookmark)
      Else
         ssBusCompInvokeMethod(ssAccount, "GotoBookmarkF", ssReport::ssBookmark)
      End If
   End If

   If (errCode = 0) And (searchSpec <> "") And ((Not bCurRowOnly) Or 
(ssReport::ssBusObjectName <> "")) Then
      ssBusCompSetSearchExpr (ssAccount, searchSpec)
   End If

   If (errCode = 0) And (SortSpecDyn <> "") Then
      ssBusCompSetSortSpec (ssAccount, SortSpecDyn)
   End If
   If (errCode = 0) And (ssReport::ssBusObjectName <> "") And (ssReport::ssViewMode <> 
"") Then ssBusCompSetViewMode (ssAccount, CInt (ssReport::ssViewMode))


   ' Re-execute business components as necessary

   If (errCode = 0) Then

      If (bReExecute) Then

        If (bCurRowOnly) And (mainRowId <> "") Then             ssBusCompSetSearchExpr 
(ssAccount, "Id = """ + mainRowId + """")

      If (errCode = 0) Then ssBusCompExecuteQuery2 (ssAccount, baseReport::bSetForward, 
True)

      Elseif (bUpdateLinks) Then

         ssBusCompInvokeMethod (ssAccount, "UpdateLinks", "")

      End If

   End If


   ' Process errors and return

   If (errCode = 0) Then

      bStatus = True

   Else

      bStatus = False

      ssProcessLastError(ssAppServer, "", "")

   End If


   Start = bStatus

End Function
```

At this point, the Siebel business component, in this case Account, has been exposed to Actuate, and the Siebel Object Manager interface has been called. The fields for the business object identified by the Siebel client’s active view have been activated and are available to the data row object.

About the Fetch Method

The Fetch method performs three functions:

1 Positions the Siebel Object Manager interface to a single row in the business object.

2. Creates a blank instance of the data row.
3. Populates the data row by calling methods on the business component Siebel Object Manager interface.

The following code sample is from the Fetch method in the ssAccountQuery datastream component in the Aclist (Account List) report.

```
Function Fetch () As AcDataRow

   Dim    bStatus     As Boolean
   Dim    bCurRowOnly As Boolean
   Dim    custDataRow As ssAccountDataRow
   Dim    theBC       As Integer

   errCode = 0
   bStatus = False
   bCurRowOnly = False
   theBC = ssAccount

   If (bCurRowOnly = True) Then
      If (Position = 1) Then
         If (baseReport::bQueryExecuted = True) Then
            bStatus = ssBusCompFirstRecord (theBC)
         Else
            bStatus = True
         End If
      End If
   Else
      If (Position = 1) Then
         bStatus = ssBusCompFirstRecord (theBC)
       If (errCode = 0) And (baseReport::bSetForward) Then ssBusCompInvokeMethod (theBC,
"SetForwardOnly", "")
      Else
```

```
         bStatus = ssBusCompNextRecord (theBC)

      End If

   End If


   If (bStatus = True) And (errCode = 0) Then

      Set custDataRow = NewDataRow


     If (errCode = 0) Then custDataRow.ssAccount_Status = ssBusCompGetFieldValue (theBC, 
"Account Status")

      If (errCode = 0) Then custDataRow.ssCity = ssBusCompGetFieldValue (theBC, "City")

      If (errCode = 0) Then custDataRow.ssCompetitor = ssBusCompGetFieldValue (theBC, 
"Competitor")

      If (errCode = 0) Then custDataRow.ssCountry = ssBusCompGetFieldValue (theBC, 
"Country")

      If (errCode = 0) Then custDataRow.ssDescription = ssBusCompGetFieldValue (theBC, 
"Description")

      If (errCode = 0) Then custDataRow.ssIndustry = ssBusCompGetFieldValue (theBC, 
"Industry")

      If (errCode = 0) Then custDataRow.ssLocation = ssBusCompGetFieldValue (theBC, 
"Location")

      If (errCode = 0) Then custDataRow.ssMain_Fax_Number = 
ssBusCompGetFormattedFieldValue (theBC, "Main Fax Number")

      If (errCode = 0) Then custDataRow.ssMain_Phone_Number = 
ssBusCompGetFormattedFieldValue (theBC, "Main Phone Number")

      If (errCode = 0) Then custDataRow.ssName = ssBusCompGetFieldValue (theBC, "Name")

      If (errCode = 0) Then custDataRow.ssParent_Account_Name = ssBusCompGetFieldValue 
(theBC, "Parent Account Name")

      If (errCode = 0) Then custDataRow.ssPostal_Code = ssBusCompGetFieldValue (theBC, 
"Postal Code")

     If (errCode = 0) Then custDataRow.ssState = ssBusCompGetFieldValue (theBC, "State")

     If (errCode = 0) Then custDataRow.ssStreet_Address = ssBusCompGetFieldValue (theBC, 
"Street Address")

      If (errCode = 0) Then custDataRow.ssSynonym = ssBusCompGetFieldValue (theBC, 
"Synonym")

      If (errCode = 0) Then custDataRow.ssType = ssBusCompGetFieldValue (theBC, "Type")
      ' Now retrieve the system fields
      If (errCode = 0) Then custDataRow.ssId = ssBusCompGetFieldValue (theBC, "Id")
      If (errCode = 0) Then custDataRow.ssCreated = ssBusCompGetFieldValue (theBC, 
"Created")
      If (errCode = 0) Then custDataRow.ssCreated_Formatted = 
ssBusCompGetFormattedFieldValue (theBC, "Created")
      If (errCode = 0) Then custDataRow.ssCreated_By = ssBusCompGetFieldValue (theBC, 
"Created By")
      If (errCode = 0) Then custDataRow.ssUpdated = ssBusCompGetFieldValue (theBC, 
"Updated")
      If (errCode = 0) Then custDataRow.ssUpdated_Formatted = 
ssBusCompGetFormattedFieldValue (theBC, "Updated")
      If (errCode = 0) Then custDataRow.ssUpdated_By = ssBusCompGetFieldValue (theBC, 
"Updated By")

      Set Fetch = custDataRow
      AddRow (Fetch)
   Else
      Set Fetch = Nothing
   End If

   If (errCode <> 0) Then
      ssProcessLastError(ssAppServer, "", "")
   End If

End Function
```

In this example, the value of bStatus defines the position of the first data row. While bStatus is true, custDataRow continues to pass rows until no values are returned. The SetForwardOnly method makes sure that all rows are processed and that duplicate rows are not passed.

NOTE: Because all data is returned as strings, two methods are called on the GetFieldValue business component Siebel Object Manager interface variable (for all data types) and on GetFormattedFieldValue (for date, currency, and other data types that require formatting).

GetFieldValue and GetFormattedFieldValue are described in Table 46.

Table 46. GetFieldValue and GetFormattedFieldValue

Method	Comments
GetFieldValue	Gets the raw data value. Dates are always mm/dd/yy hh:mm:ss. Numbers and currency are always strings. Can be used to sum numeric values.
GetFormattedFieldValue	Uses the format specified in the Siebel client. Called by default for each currency field (“$1,234.34”). Called by default for any date field (“01/12/31”). Numeric values cannot be summed.

NOTE: These calls to the two methods are no longer done directly. Instead, these are called as ssBusCompGetFieldValue and ssBusCompGetFormattedFieldValue. These methods are implemented in sssiebel.bas and call the above mentioned methods.

About the Delete Method

The Delete method destroys the datastream object and frees system resources. In the example below, custom code has been added before the Delete method on the superclass is called. Note that ssAcount has handled the deletion duties for its child datastreams and has handed the calls back to the server (EnableNotify).

The following code sample is from the Delete method in the ssAccountQuery datastream component in the Aclist (Account List) report.

```
Sub Delete ()


   ssRestoreActiveRow (ssAccount)

   If Not (ssAccount = 0) Then ssBusCompInvokeMethod (ssAccount,
"SetIgnoreMaxCursorSize", "N")

   If Not (ssAccount = 0) Then ssBusCompAllowNotification (ssAccount)

   ssAccount = 0


   Super::Delete ()

End Sub
```

Reports frequently have multiple report sections nested within one another to display detail, summary, or related data. Each report section requires a separate datastream component, and it is desirable that the datastreams be nested as well. This allows the master datastream to perform Start and Delete tasks for the child, enhancing efficiency.

The roles of the master datastream are:

- To create the Siebel Object Manager interface handling calls to the server
- To activate all fields for itself and its children
- To decide whether a query needs to be executed again in order to accommodate a sort specification
- To delete itself and all its children

The roles of the child datastream are:

- To fetch DataRows
- To format the fields as required
- To perform sort specifications

NOTE: Master datastreams do not have numbers appended to their names.

C Smart Reports List of Values

This appendix summarizes the List Of Values used in Smart Reports and indicates how they are currently used in the graphical elements. For more information see, Chapter 12, “Smart Reports.”

The following tables list the List Of Values used by certain Smart Reports.

Opportunity Detail Report

Table 47 lists the List Of Values used with the Opportunity Detail report.

Table 47. Opportunity Detail Report List Of Values

Graphical Element	List of Values Used	Thermometer Variable	Calculation Point	Comment
Buying Influencers Thermometer	NA NA NA NA	Trigger Value Max Value Min Value Data Value	ssList_Of_Values Query3	The weighted average score is calculated based on LOV types CONTACT_ROLE, TAS_POLITICAL_ ANALYSIS, and TAS_ORG_STATU S.
Probability Thermometer	Order_By Target_High Target_Low Rep_	Trigger Value Max Value Min Value Data Value	Content - Probability Thermometer (under Content - frmProbability)	The probability of the opportunity in Rep_field of the opportunity displayed.
Deal Size Thermometer	Order_By Target_High Target_Low Functional_Revenue	Trigger Value Max Value Min Value Data Value	Content - Deal Size Thermometer (under rptDealSizeThermometer)	The LOV values associated with Functional Revenue field are used.

Account Service Detail Report

Table 48 lists the List Of Values used with the Account Service Detail report.

Table 48. Account Service Detail Report List of Values

Graphical Element	List Of Values Used	Thermometer Variable	Calculation Point	Comment
Revenue Thermometer	Order_By Target_High Target_Low totalRevenue	Trigger Value Max Value Min Value Data Value	dsRevenueLOV dsRevenueLOV dsRevenueLOV sifAllOpportunities	Total Revenue for the account relative to the average of across all accounts is shown.
Open Service Request Thermometer	avgOpenSRs 0 countOpenSRs	Trigger Value Max Value Min Value Data Value	dsTargetOpenSrsLOV Local Local sifAllServiceRequests	The count of the open service requests for this account relative to the average across all accounts is shown.
Customer Satisfaction Thermometer	Order_By Target_High Target_Low NA	Trigger Value Max Value Min Value Data Value	ssList_Of_Values Query1 ssList_Of_Values Query1 ssList_Of_Values Query1 Local	The ratio of total customer satisfaction score to the total number of surveys shown.

Account Summary Report

Table 49 lists the List Of Values used with the Account Summary report.

Table 49. Account Summary Report List of Values

Graphical Element	List Of Values Used	Thermometer Variable	Calculation Point	Comment
Past Revenue Thermometer	Order_By Target_High Target_Low totalRevenue	Trigger Value Max Value Min Value Data Value	qryPastRevenueListOfValues qryPastRevenueListOfValues qryPastRevenueListOfValues qryPastRevenue	Sum of revenues for the opportunities associated with an account is displayed.
Pipeline Thermometer	Order_By Target_High Target_Low Pipeline Revenue	Trigger Value Max Value Min Value Data Value	qryPiplineListOfValues ftrPipelineThermometer	Pipeline revenue for all opportunities associated with an account is displayed.
Customer Satisfaction Thermometer	Order_By Target_High Target_Low NA	Trigger Value Max Value Min Value Data Value	qryCustomerSatisfactionList_Of_Values (all)	The ratio of total customer satisfaction score to the total number of surveys is shown.

Pipeline Analysis Report

Table 50 lists the List Of Values used with the Pipeline Analysis report.

Table 50. Pipeline Analysis Report List Of Values

Graphical Element	List Of Values Used	Thermometer Variable	Calculation Point	Comment
Revenue versus All Current Quotas Thermometer	totalQuota	Trigger Value	sifQuota	Sum of revenues for the next four quarters is displayed. Also displayed is a dashed line below the trigger that indicates a revenue level that is acceptable, but below the target.
	(Calculated)	Acceptable Range	dsTargetQuotaRange	
	(Calculated)	Max Value	Local	
	0	Min Value	Local	
	ExpectedRevenueNext4Q	Data Value	frmDashboard	

D List of Selected Reports

NOTE: An asterisk (*) represents those reports that were redesigned in Siebel 7.5. Two asterisks (**) represent reports new in Siebel 7.5.

The reports, with navigational paths, included in this list are those available across various modules in Siebel 7.x. Some of these reports may be removed and new reports added in later versions of the Siebel application. Also, the functionality of the reports listed may be modified in future releases.

Template Name refers to the report design file name in ...\Tools\RPTSRC\ENU\STANDARD. Parameterized reports are clearly marked.

Table 51 contains a listing of reports in the Siebel eBusiness Application that can be modified using Actuate.

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Accounts - Current Query	ACLIST	Account List	Lists all the accounts		Accounts > My Accounts
Account Summary	ACSUM	Smart Report Account Summary	Smart Report - Describes all details about the account graphically		Accounts > My Accounts
Account Service Detail	ACSVCDET	Smart Report Account Service Detail	Smart Report - Summarizes all the service-related info. about the account graphically		Accounts > My Accounts
Account Service Profile	ACSVCPRO	Account Service Profile	Report list all the accounts' service profiles		Accounts > My Accounts
Activity - Current Query	ACTLIST	Activity List	Lists all the activities		Activity > My Activities
Admin Competitors List - Current Query	ADCOMP	Admin Competitor List	Marketing administration competitors list		Data Administration > Competitor

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Admin Decision Issues - Current Query	ADDECIS	Admin Decision Issue List	Marketing administration decision issues list		Data Administration > Decision Issues
Admin Client - Current Query	ADLIST	Mobile Client List	Siebel Remote Admin Mobile Client List		Siebel Remote Administration > Replication Servers
Admin Literature - Current Query	ADLIT	Admin Literature List	Marketing administration literature list		Document Administration > Literature
Admin Products - Current Query	ADPROD	Admin Product List	Marketing administration product list		Product Administration
Admin Product Line - Current Query	ADPRODLI	Admin Product Line List	Marketing administration product line list		Application Administration > Product Lines
Admin Sales Cycle - Current Query	ADSLSCYC	Admin Sales Cycle List	Siebel Assistant Admin Sales Cycle List		Application Administration > Sales Methods
Agreement Detail	AGDET	Current Agreement Detail	List of all agreements with details		Agreements
Agreement Summary	AGSUM	Agreement Summary	List of summaries of all agreements		Agreements > My Agreements
Agent Performance Detail	AGTPERDET	Employee Performance Detail	List all Open Service Requests, group by Employee Login, with one “Severity” pie chart per login		Service > My Team's Service Request
Agent Roles and Responsibilities	AGTROLRES	Employee Roles and Responsibilities	List of Employees, with their Skills and Skill Items		User Admin > Employees

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
ePricer - Account Specific Price Book	APRICEBOOK	Account Specific Price Book	Lists all quotes and their accounts		Quotes
Assignment Manger Detail	ASMANDET	Assignment Manager Detail	Assignment Manager Detail Report		Assignment Administration > Assignment Rules
DCommerce: Customer List Report	AUCCUSLST	Customer List Report	Displays list of customers	Yes	eSales > eAuction home page > Select Reports Link from Lister Area
DCommerce: Auction Detail Report	AUCDET	Auction Detail Report	Displays all details about Auction	Yes	eSales > eAuction home page > Select Reports Link from Lister Area
DCommerce: Auction/RFQ Overview Report	AUCRFQSUM	Auction/RFQ Overview Report	Displays RFQ overview data	Yes	eSales > eAuction home page > Select Reports Link from Lister Area
DCommerce: Runner's up Report	AUCRUNUP	Runner's up Report	Displays list of losers	Yes	eSales > eAuction home page > Select Reports Link from Lister Area
DCommerce: Winners Report	AUCWIN	Winners Report	Displays list of winners	Yes	eSales > eAuction home page > Select Reports Link from Lister Area

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
FS Below Minimum Inventory Quantity Per Location	BELOWMIN	Products Below Minimum Level By Location	Products (inventory quality) below minimum level by location		Inventory
DCommerce: Bid List By Customer	BIDLSTCUST	Bid List By Customer	Displays bids for a Customer	Yes	eSales > eAuction home page > Select Reports Link from Lister Area
DCommerce: Bid List By Auction	BIDLSTLOT	Bid List By Auction	Displays bids for an Auction	Yes	eSales > eAuction home page > Select Reports Link from Lister Area
Business Service Detail (Client)	BUSSVCDETC	Business Service Detail	List of all business services with details		Business Service Administration
Business Service Summary (Client)	BUSSVCSUMC	Business Service Summary	List of summaries of all business services		Business Service Administration
Campaign - Current Query	CAMPLIST	Campaign List	List of all campaigns		Campaigns > My Campaigns
Campaign Response Detail	CAMPRESPDET	Campaign Response Detail	List of all responses with details		Response > My Responses
Campaign Response Summary	CAMPRESPSUM	Campaign Response Summary	Summarizes all responses (sort by campaigns)		Response > My Reponses
Campaign Summary	CAMPSUM	Campaign Summary	Summary of all campaigns		Campaigns
eChannel - CHAMP Partner Plan	CHAMPPLAN	CHAMP Partner Plan	Displays plan data for current partner		Partners

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Calendar - Daily*	CLDAY	Current Day	Lists all activities for current day		Calendar
Calendar - Monthly*	CLMON	Current Month	Lists all activities for current month		Calendar
Calendar - Weekly*	CLWEEK	Current Week	Lists all activities for current week		Calendar
Contacts By Category	CNTCAT	Contacts By Category	Lists all contacts by category	Yes	Contacts
Contacts - Current Query	CNTLIST	Contact List	List of all contacts		Contacts
Contact - Current Query (Personal)	CNTLISTPER	Contact List	List of all personal contacts		Contacts > Personal Contacts
Contacts - By Opportunity	CNTOPP	By Opportunity	List of all contacts (sort by opportunities)		Contacts
Contacts - Alphabetic Phone List	CNTPHON	Alphabetic Phone List	A phone list of all contacts in alphabetical order		Contacts
Contacts - Alphabetic Phone List (Personal)	CNTPHONPER	Alphabetic Phone List	A phone list of all personal contacts in alphabetical order		Contacts > Personal Contacts
eTraining - Curriculum Detail	COCURR	Curriculum Detail	Curriculum detail report		Training Administration > Curriculums
eTraining - Course List	COLIST	Course List Report	eTraining Course List Report		Training Administration > Course Lists
Competitors List - Current Query	COMPLIST	Competitor List	List of all competitors		Competitors

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Compensation Statement	COMPSTAT	Compensation Statement	Compensation Statement		Compensation
eTraining - Course Registration	COREGIS	Course Registration Report	Course registrant report		Training Administration > Course Lists
eTraining - Curriculum Registrant	COREGNT	Curriculum Registrants	Curriculum registrant report		Training Administration > Curriculums
Correspondence - Current Query	CORESP	Correspondence Request List	Correspondence Request List		Correspondence
eTraining - Course Schedule	COSCHED	Course Schedule Report	Course schedule report		Training Administration > Course Lists
eTraining - Skills Test Detail	COSKEXAM	Skills Test Detail	Skills examination detail report		Training Administration > Test > Test Question
PS - Combined Time Expense Summary*	COTESUM	Combined Time and Expense Summary	List of expenses involved with all projects		Projects > All Projects
eTraining - Training Library List	COTRLIB	Training Library List	Training library report		Training Administration > Training Library
Call Center Volume	CSVOLUME	Call Center Volume	Generate spreadsheet and pie charts of both Inbound and Outbound communication channels		Communication Administration > Report
Call Center Volume (Inbound)	CSVOLUMEIN	Call Center Volume (Inbound)	Generate spreadsheet and pie charts of Inbound communication channels		Communication Administration > Report

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Call Center Volume (Outbound)	CSVOLUMEOUT	Call Center Volume (Outbound)	Generate spreadsheet and pie charts of Outbound communication channels		Communicatio n Administration > Report
Current Account Service Profile	CUACCSVC	Current Account Service Profile	List of service for current account		Accounts > My Accounts
Current Siebel Remote Session	CUDOCSES	Current Siebel Remote Session	Displays the status of current remote session		User Preferences > Remote Status
FS - Customer Invoice	CUSTINV	Customer Invoice	Reports customer invoices		Invoices
FS Cycle Count Detail	CYCCOUNTDET	Cycle Count Detail	Reports all product cycle counts with details		Cycle Counts
Decision Issues - Current Query	DESISS	Decision Issue List	List of all decision issues		Decision Issues > All Decision Issues
EC Expense Report	ECEXPREP	Euro Expense Report	Euro Expense Report		Expense Reports > My Expense Reports
Employee Achievement Report	EMPACH	Employee Achievement	Lists all quota achievements for the employees		Compensation > Quota Achievement
Employee List - Current Query	EMPLIST	Employee List	List of all employees		User Administration > Employees
PS - Employee Staffing Schedule	EMPSCHED	Employee Schedule	Displays data for each employee		User Administration > Employees > Employee Availability

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
PS - Employee Time Expense Summary*	EMPTESUM	Employee Time and Expense Summary	List of expenses involved with a project for the employee		Application Administration > Employees > Employee Availability
PS - Employee Utilization	EMPUTL	Employee Utilization	Lists workable hours versus Total hours and calculates utilization %		User Administration > Employees > Employee Availability
Release - Engineer Task Detail	ENGTASKDET	Engineer Task Detail Summary	Lists engineering tasks and details		Release > Engineering Tasks
All Projects Overview**	ERMPROJOV	All Projects Overview	Reports an overview of all projects		Projects > All Projects
Workflow Log	ESCLOG	Workflow Log	Reports the Workflow Log for Workflow Administration		Siebel Workflow Administration > Workflow Policy Log
Workflow Policy	ESCPOL	Workflow Policy	Reports workflow policies for Workflow Administration		Siebel Workflow Administration > Workflow Policies
ESP Account Plan Overview Report	ESPCOV	Account Plan Overview	ESP page report. Shows Account/BU objectives in a page format		Account > ESP > Account Plan Overview
ESP Account Map Report*	ESPSS	ESP Account Map	ESP Account Map: Spreadsheet report. Shows crosstab BusUnits versus Offerings		Account > ESP > Account Plan Overview

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Expense Summary Report - Accounts	ESRACC	Expense Summary By Account	Summarizes all expenses by accounts		Expense Reports > My Expense Reports
Expense Summary Report - Charge Number	ESRCHARGE	Expense Summary By Charge #	Summarizes all expenses by charge numbers		Expense Reports > My Expense Reports
Expense Summary Report - Opportunities	ESROPP	Expense Summary By Opportunity	Summarizes all expenses by opportunities		Expense Reports > My Expense Reports
Event Status Report	EVENTSTAT	Event Status	Displays event status		Events
Event Summary Report*	EVENTSUM	Event Summary	Summary of events		Events
Expense Report	EXPREP	Expense Report	Report of all expenses		Expense Reports > My Expense Reports
FS Field Engineer Detail	FEADET	Field Engineer Activity Detail	Reports detailed field engineer activities		Activities > All/My Activities
FS Field Engineer Activity Summary	FEASUM	Field Engineer Activity Summary	Summarizes field engineer activities		Activities > All/My Activities
Release - Feature Detail	FEATDET	Feature Detail Summary	Summary of feature details		Release > Features
Forecast Analysis Detail	FODET	Forecast Analysis Detail	Forecast Revenue Spreadsheet detailed by products	Yes	Forecast > Forecast Detail

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Forecast Analysis Summary	FOSUM	Forecast Analysis Summary	Forecast Revenue Spreadsheet aggregated	Yes	Forecast > Forecast Detail
eChannel - Fund Request Summary	FUNDREQSUM	Fund Request Summary	Lists all fund requests		Fund Request
IC - Compensation Groups	ICCGROUPS	Compensation Groups	Displays Incentive Compensation Groups		Incentive Compensation Admin > Compensation Groups
IC - Manager Summary	ICMSUM	Manager Compensation Summary	Displays compensation summary and revenue analysis for the team (direct reports)		Compensation > My Team's Compensation
IC - Employee Sales Compensation **	ICPERPLAN	Employee Personal Sales Compensation Plan	Communicates sales compensation plans to the sales representative s		Incentive Compensation Administration > Participant > Plan > Incentive Compensation Plan Participant Component View
IC - Participant Groups	ICPGROUPS	Participant Groups	Displays list of participating compensation groups for each participant		Incentive Compensation Admin > Plans
IC - Plan Definition	ICPLANDEF	Plan Definition	Displays compensation plan components		Incentive Compensation Admin > Plans

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
IC - Rep Summary	ICRSUM	Representative Order Summary	Displays revenue and order summary analysis for a Sales Representative		Compensation > All/My Compensation
Incentive Component Payout	INCOMPAY	Incentive Compensation Component	Compensation summary of the representative		Incentive Compensation Tracking > Plan Rule Payout
FS - Inventory Cost Detail	INCOSTDET	Inventory Cost Detail	Displays field service inventory cost detail		Inventory > My/All Inventory Location > Reports
Literature - Current Query	LITCUR	Literature List	List of all literatures		Literature
Literature Fulfillment	LITFUL	Literature Fulfillment	Reports all literature fulfillments		Fulfillment
eChannel - Marketing Funds Detail	MKTFUNDDET	Market Development Funds Detail	List all funds		Fund design
Release - MRD Detail	MRDDET	MRD Detail Summary	Summary of MRD details		Release > Marketing Requirement Documents
Message List	MSGLIST	Message List	List of Messages		Messages
Opportunities - By Category	OPCAT	Opportunities By Category	List of Opportunities by category	Yes	Opportunities
Opportunity Detail	OPDET	Smart Report - Opportunity Detail	A detailed list of all opportunities		Opportunities

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Opportunity List - Current Query	OPLIST	Opportunity List	Lists all opportunities		Opportunities
Opportunity Marketing Events Summary Reports**	OPMRKTSUM	Opportunity Marketing Events Summary	Number of opportunities that are generated/ accepted by each category of Marketing events and by region		Opportunities > My Opportunities
Opportunities - By Sales Rep	OPSLSREP	By Sales Rep	Lists all opportunities sorted by sales representative s		Opportunities
Opportunity Status Report**	OPSTATUS	Opportunity Status	Total number of opportunities created in the current calendar or fiscal quarter by status		Opportunities > My Opportunities
Opportunities - Summary	OPSUM	Opportunity Summary	Smart Report - Summarizes all opportunities graphically		Opportunities
Order Detail	ORDET	Order Detail (Barcode)	A detailed list of current order		Orders
Orders Detail	ORDETALL	All Orders Detail	A detailed list of all orders (barcoded)		Orders
Order Detail - No Barcode	ORDETNBC	Order Detail (No Barcode)	A detailed list of all orders (no barcode)		Orders
Orders Summary	ORSUMALL	All Orders Summary	Summarizes all orders		Orders

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
eChannel - Partner Operations	PARTOP	Partner Operations	Channel partner operations list		Partner Operations
eChannel - Partner Profile	PARTPRO	Current Partner Profile	List of all data for a current partner		Partners
Partner List	PARTRLIST	Partner List	List of partners		Partners
eChannel - Partner Report Card	PARTRPTCARD	Partner Report Card	Displays graphs based on objectives for current partner		Partners
FS Pick Ticket Details	PICKTICKET	Pick Ticket Details (Barcode)	Reports details of pick tickets barcoded (shipping)		Shipping
FS Pick Ticket Details - No Barcode	PICKTICKETNBC	Pick Ticket Details (No Barcode)	Reports details of pick tickets no barcoding (shipping)		Shipping
Pipeline Analysis	PIPEANA	Smart - Report Pipeline	Smart Report - Measures performance relative to all active quota plans graphically		Opportunities > My Team's Opportunities
Pipeline Report By Rep.	PIPEREP	Pipeline Report By Rep	Pipeline Report by Representative (Opportunities)		Opportunities
PS - Project Labor Burn Rate	PLBURNRATE	Labor Burn Rate	Lists all projects and calculates actual and forecasted burn rates for each project		Projects

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
ePricer - Price List based Price book generation	PLPRICEBOOK	Price List Based Price Book	Displays pricing information in a price list		Pricing Administration
Position List - Current Query Group	POSLSTCQ	Position List	List of positions (Application Administration)		Administration > Positions
Price List - Current Query	PRICELST	Price Lists	Price List		Pricing Administration > Price List
Pricing Factors - All	PRICERFACTA	Pricing Factors - All	List of all pricing factors		Pricing Administration > Pricing Manager > Pricing Factor Designer
Pricing Factors - Single	PRICERFACTS	Pricing Factors - Single	Current pricing factor		Pricing Administration > Pricing Manager > Pricing Factor Designer
Pricing Factors	PRICINGFACT	Pricing Models - All	List of all pricing models and their factors		Pricing Administration > Pricing Manager
Products - Current Query	PRODCQ	Product List	List of all products		Products
Product Defect Activity	PRODEFAC	Change Request Activity	Lists all activities associated with change requests		Quality > Change Request List
Product Defect - Current Query	PRODEFQ	Product Defect Summary	Product Defect Summary		Quality > All Change Requests

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Product List	PRODLIST	Product List	Lists all products (Service Inventory)		Products
FS Product List By Location	PRODLISTLOC	Product List By Location	Lists all products by location (Service Inventory)		Inventory
Project Status Report	PROJSTAT	Project Status	List of all projects and their status		Projects > My/ All Projects
PS - Project Limits	PROLMT	Project Limits	Lists all time expense limitation for the project		Projects
PS - Project Time and Expense Summary Summary	PRTESUM	Project Time and Expense	List of expenses involved with a project		Projects
Quota Summary	QUOSUM	Smart Report - Quota	Smart Report - Measures performance relative to all active quota plans graphically		Opportunities > My Team's Opportunities
Quote - Current Query	QUOTECQ	Quote List	Quote List		Quotes
Quote Summary	QUOTENPKG	Summary Quote	Summary of Quote		Quotes
Quote with Packages	QUOTEPKG	Package Quote	Summary of quote with packages		Quote
Proposal Quote	QUOTEPRO	Proposal Quote	Defined for use by Proposal Generator. Will not appear in a view		Available only with Proposal Generator (not in views)

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Standard Quote	QUOTESTD	Current Quote	Standard quote report		Quotes
FS Repair Detail	REPDET	Repair Detail (Barcode)	Detailed list of Repairs (barcoded)		Repairs
FS Repair Detail - No Barcode	REPDETNBC	Repair Detail (No Barcode)	Detail list of Repairs (no barcoding)		Repairs
FS Repair Summary	REPSUM	Repair Summary	Summary of Repairs		Repairs > All Repairs
Revenue Analysis Detail	REVDET	Revenue Analysis Detail	Revenue Spreadsheet detailed by products	Yes	Revenues
Revenue Analysis Summary	REVSUM	Revenue Analysis Summary	Revenue Spreadsheet aggregated	Yes	Revenues
Service Request Activity - All	SRVREQAA	Service Request Activity (All)	List of all service requests		Service > My Service Requests
Service Request Detail	SRVREQDT	Service Request Detail (Barcode)	Lists service requests with details (barcoded)		Service > My Service Requests
Service Request Detail - No Barcode	SRVREQDTNBC	Service Request Detail (No Barcode)	Lists service requests with details (no barcoding)		Service > My Service Requests
Service Request Activity - Public	SRVREQPA	Service Request Activity (Public)	List of all public service requests		Service > My Service Requests
Service Request Summary	SRVREQSM	Service Request Summary	Summarizes service requests		Service > My Service Requests

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Service Request Aging Analysis	SRVRQAGAN	Smart Report - Service Request Performance	Smart Report - Analyzes the aging of currently open service requests graphically		Service > My Service Requests
Service Status By Channel	SRVSTATCH	Service Status by Channel	Bar chart and table with counts of all open Service Requests with Channel: Fax, Email, Web, or Phone		Service > My Team's Service Request
Release - Strategy Detail	STRATEGYDET	Test Strategy Detail Summary	Summary of test strategy details		Release > Strategy
Opportunities - Strategic Selling	STRATSEL	Blue Sheet	Strategic Selling		Opportunity > Strategic Selling
PS - Subcontractor Cost Margin Rate	SUBCOMR	Subcontractor Cost and Margin	Lists expenses for the Subcontractor		Projects > All Projects > Subcontractor s > Account Field > Subcontractor Employee
TAS - Initial Plan*	TASIPLAN	TAS Initial Plan	TAS - Initial Plan (Opportunities)		Opportunity > TAS (all views)
TAS - Opportunity Plan*	TASOPLAN	TAS Opportunity Plan	TAS - Opportunity Plan (Opportunities)		Opportunities > TAS > Plan
Release - Tech Doc Detail	TECHDET	Technical Publications	Summary of technical publications details		Release > Tech Doc

Table 51. Listing of Reports in Siebel eBusiness Application

Report Name	Template Name	Menu Text	Description	Parameter Report	Navigation Path
Territory Assignment Detail	TERASDET	Territory Assignment	Reports Territory Assignment with details (Assignment Administration)		Assignment Administration
Release - Test Detail	TESTDET	Test Detail Summary	Summary of test details		Release > QA Test
Release - Test Plan Detail Summary	TESTPLDET	Test Plan Detail	Summary of test plan details		Release > QA Test Plan
PS - New Time Sheet	TIMESH	Time Sheet	Lists the time sheet		Time Sheets
Test Plan Defect Summary	TRACKDEFECT 2000	Test Plan Defect Summary	Summary of test plan defects		Release > QA Test
Client Stability Status Execution	TRACKSTAB STATUS2000	Test Plan Execution Summary	Summary of test plan execution		Release > QA Test
Client Stability Status Summary	TRACKSTATUS 2000	Client Stability Summary	Summary of Client stability		Release > QA Test
Server Stability Status Summary	TRACKSTATUS 2000	Server Stability Status Summary	Summary of Server stability		Release > QA Test

E Synchronizing Locale-Sensitive Parameters

For the Reports Server to format reports based on a user-specified locale, the locale definitions in localemap.xml file should match the equivalent parameters in the Siebel Application Object Manager.

For more information about setting parameters for application object managers, see the *Siebel System Administration Guide*.

Verify that the locale-specific parameters in the localemap.xml file match the corresponding parameters in the Application Object Manager. See the *Global Deployment Guide* for a list of these parameters.

Table 52. Windows Regional Settings and Application Object Manager Parameters

Tab in Regional Settings Dialog Box	Windows Parameter	Application Object Manger Parameter	Description
Number	Decimal symbol	`sDecimal`	Character used to separate the integer part from the fractional part of a number and currency.
	Digit grouping symbol	`sThousand`	This is the symbol used to separate thousands in numbers and currencies with more than three digits.
	Number of Digits after Decimal	`iDigits`	Value defining the number of decimal digits that should be used in a number.
	Display leading zeros	`iLzero`	`iLzero=0` > Display without leading zero. `siLzero=1`> Display with leading zeros.

Table 52. Windows Regional Settings and Application Object Manager Parameters

Tab in Regional Settings Dialog Box	Windows Parameter	Application Object Manger Parameter	Description
Currency	Decimal symbol	sDecimal	Character used to separate the integer part from the fractional part of a number and currency.
	Digit grouping symbol	sThousand	This is the symbol used to separate thousands in numbers and currencies with more than three digits.
		iCurrency	iCurrency=0 > no separation between currency symbol prefix and number. iCurrency=1 > no separation between currency symbol suffix and number. iCurrency=2 > one character separation between currency symbol prefix and number. iCurrency=3 > one character separation between currency symbol suffix and number.

	Negative currency format	iNegCurr	iNegCurr=0 -> ($1.1)
			iNegCurr=1 -> -$1.1
			iNegCurr=2 -> $-1.1
			iNegCurr=3 -> $1.1-
			iNegCurr=4 -> (1.1$)
			iNegCurr=5 -> -1.1$
			iNegCurr=6 -> 1.1-$
			iNegCurr=7 -> 1.1$-
			iNegCurr=8 -> -1.1 $
			iNegCurr=9 -> -$ 1.1
			iNegCurr=10 -> 1.1 $-
			iNegCurr=11 -> $ 1.1-
			iNegCurr=12 -> $ -1.1
			iNegCurr=13 -> 1.1- $
			iNegCurr=14 -> ($ 1.1)
			iNegCurr=15 -> (1.1 $)

Table 52. Windows Regional Settings and Application Object Manager Parameters

Tab in Regional Settings Dialog Box	Windows Parameter	Application Object Manger Parameter	Description
Parameter	Time separator	sTime	Time separator. This character is displayed between hours and minutes, and between minutes and seconds.

	Time style	iTime	iTime=0 > 12-hour clock.
			iTime=1 > 24-hour clock.

	Time style	iTLZero	Specifies whether or not the hours should have a leading zero.
			iTLZero=0 > without leading zero.
			iTLZero=1 > with leading zero.

	AM symbol	s1159	This setting contains the trailing string used for times between 00:00 and 11:59.
	PM symbol	s2359	Trailing string for times between 12:00 and 23:59, when in 12-hour clock format.
Date	Date separator	sDate	Character used to separate the integer part from the month, day and year either using a slash (/) or a dash (-).
	Short date style	sShortDate	Date value in the format mm/dd/yy or mm-dd-yy where the month (mm), day (dd), and year (yy) are expressed as two-digit numbers.
	Long date style	sLongDate	Date value in the format mm/dd/yyyy or mm-dd-yyyy where yyyy represents the year expressed as a four-digit number.

Index