SIEBEL 7
eBusiness

SIEBEL SMARTSCRIPT ADMINISTRATION GUIDE

VERSION 7.5.3

12-FRKILG

JUNE 2003

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

Printed in the United States of America

Contents

Chapter 3. Working with Questions, Answers, and Translations

Chapter 4. Upgrading to SmartScript 7.5.3

Chapter 5. Customizing a SmartScript User Interface

Chapter 6. Verifying, Testing, and Invoking SmartScripts

Chapter 7. Extending Scripts with Siebel VB and Siebel eScript

Introduction

This guide provides an overview of Siebel SmartScript and its use from an employee perspective. The primary focus of this guide is to provide configuration and administration instructions to allow you to set up Siebel SmartScript on your Siebel application.

Although job titles and duties at your company may differ from those listed in the following table, the audience for this guide consists primarily of employees in these categories:

Call Center Administrators	Persons responsible for setting up and maintaining a call center. Duties include designing and managing Computer Telephony Integration (CTI), SmartScripts, and message broadcasts.
Database Administrators	Persons who administer the database system, including data loading, system monitoring, backup and recovery, space allocation and sizing, and user account management.
Marketing Administrators	Persons responsible for setting up and maintaining a marketing department. Duties include designing and managing campaigns, product marketing information, and product distribution lists.
Siebel Application Administrators	Persons responsible for planning, setting up, and maintaining Siebel applications.
Siebel Application Developers	Persons who plan, implement, and configure Siebel applications, possibly adding new functionality.
Siebel System Administrators	Persons responsible for the whole system, including installing, maintaining, and upgrading Siebel applications.

Product Modules and Options

This Siebel Bookshelf 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 Bookshelf. To find out more about the modules your organization has purchased, see your corporate purchasing agent or your Siebel sales representative.

How This Guide Is Organized

This guide provides information necessary to implement, configure, and administer Siebel SmartScript in your Siebel application.

- Read Chapter 1, "Overview," for an introduction to SmartScript.
- Read Chapter 2, "Fundamentals of SmartScript Creation," and Chapter 3, "Working with Questions, Answers, and Translations," for information on setting up and Administering SmartScript.
- Read Chapter 4, "Upgrading SmartScripts," for information on upgrading previous SmartScript versions to Version 7 of Siebel SmartScript.
- Read Chapter 5, "Customizing a SmartScript User Interface," and Chapter 9, "Modifying the Dashboard," if you are going to customize the look and feel of your SmartScript user interface.
- Read Chapter 6, "Verifying, Testing, and Invoking SmartScripts," and Chapter 7, "Extending Scripts with Siebel VB and Siebel eScript," and Chapter 8, "Importing and Exporting SmartScripts," for information on deploying your SmartScripts and further enhancing them with Siebel VB and eScript.

Revision History

Siebel SmartScript Administration Guide

Version 7.5.3

Table 1. Changes Made in Version 7.5.3

Topic	Revision
"Releasing Scripts"	Revised for 7.5.3: Added overview of release process.
"Formatting Question Text Using HTML Tags"	Revised for 7.5.3: Added text about Information label text and the way in which you customize this text.
"SmartScript Diagnostics"	New for 7.5.3: Added new topic.
"SetCurrentValue"	Revised for 7.5.3: Added to definition.

Version 7.5, Rev A

Table 2. Changes Made in Version 7.5, Rev A

Topic	Revision
"The Employee's Screen"	Added discussion of functionality of SmartScript buttons in employee user interface.
"Displaying Answers Within a SmartScript"	Added note indicating that SmartScripts cannot be configured to handle association applets.
"Data Storage in SmartScript"	Added more information about Save Session parameter.
"Overview of Migration from Pre-Siebel 7 Releases of SmartScript"	Added a note indicating that browser scripts are not supported in SmartScripts.
"SetQuestionEnable"	Added Siebel eScript example.

Additional Changes

Added new heading structure to make HTML navigation easier.

1 Overview

Siebel SmartScript allows business analysts, call center managers, and Siebel developers to create scripts to define the application workflow for interactive customer communications. The flow of the interaction is determined entirely by the script, not by the agent or customer.

Siebel SmartScript guides agents through each customer interaction, suggesting products and services based on the customer's profile, environment, current requirements, and buying patterns. SmartScript helps agents overcome customer objections, address competitive issues, and, in service calls, ask the right questions to resolve problems. Escalations and fulfillments are routed automatically. Department administrators can create business process workflows and scripts and then change them in real time to improve productivity without interrupting call center operations.

SmartScript can be used for customer applications, allowing the same troubleshooting scripts to be shared between customers using the Siebel eService site, and call center agents using Siebel Call Center.

Traditional communications include inbound interactions, where a customer calls a customer service agent and receives information about loan processing, or a service issue. Outbound interactions involve agents contacting customers, such as telemarketing.

Siebel eBusiness customer communications include eMarketing applications where customers interact with Web surveys, or email questionnaires, or customers using interactive troubleshooting guides on Service Web sites.

Benefits of SmartScripts

Siebel SmartScript offers these fundamental benefits:

- **Software-controlled workflow.** SmartScript enforces the business processes of the enterprise by means of a script that the call center agent or customer must follow. The script guides the agent or customer through each step of the appropriate business process, typically by providing a sequence of questions. SmartScript selects the appropriate branch of questions as needed. Branching activity is based on the answers and responses selected by the customer or prospect.
- **Reduced training time.** SmartScript guides even inexperienced users through a set process. Users are prompted with what questions to answer or ask, and what information to read.
- **Simple workflow design and implementation.** SmartScript allows business analysts and call center managers, rather than systems analysts or programmers, to design and implement the workflow. This brings first-hand knowledge to process definition, allowing business experts to build SmartScripts question by question.
- **Intuitive graphical user interface.** SmartScript Designer is a visual tool that allows a script administrator to create scripts by graphically manipulating script elements to define a workflow. Technical programming skills are not required; however, an analyst who does have programming skills can use Siebel VB or Siebel eScript to enhance and extend the capabilities of SmartScripts.
- **Personalized Interaction.** Both the questions that are asked, and the logic of the script can be adjusted based on customer information or on answers provided previously in the script. This allows each experience using the script to be personalized and effective.
- **Dynamic updating.** Using branching logic, SmartScript displays only those questions in a script that are pertinent to a given transaction.
- **Search Center integration.** SmartScripts may now be searched against and displayed in the Search Center results field. The user may then click on the hyperlink and launch the SmartScript.

- **Dashboard.** Information gained from a SmartScript can be dynamically updated to the customer dashboard so it can be viewed during the course of the customer interaction.
- **Efficient modification and reuse of scripts.** Scripts are built of modular elements: questions, pages of questions, answers, branching instructions. An element, such as a page of questions—or even an individual question—can be used in multiple scripts or in multiple language translations for a single script.

Prerequisites

To use SmartScript effectively, you should fulfill the following prerequisites:

- Familiarity with Siebel user interface standards.
- If you want to use advanced scripting to extend the standard capabilities of SmartScript, you need proficiency in Siebel VB or Siebel eScript programming and knowledge of your company's Siebel installation. Siebel VB (Visual Basic) and Siebel eScript (a JavaScript-like scripting language that shares tools with Siebel VB) can be used with all SmartScript elements.
- If you want to deploy SmartScript within a multilingual call center, you need translations for all script elements in each language in which the script will be run.
- If you want to deploy graphically customized SmartScripts, such as SmartScripts that are graphically integrated with existing Web sites, you need HTML editing and Web site design experience.

Scripting Terminology

You should be familiar with the terms listed in Table 1 to understand scripting.

Table 1. Scripting Terminology

Term	Definition
element	Any named part of a script, such as a question, an answer, a page, or a branch.
script	The object that contains all subsidiary content and procedural elements for directing the workflow for an interaction. It consists of a name and a collection of pages, plus the branches needed to move between pages.
page	Logical grouping of questions within a page which will display together to the user.
question	Script element that is a question to be asked of customers, or a text message that provides information to the agent. Questions are displayed on pages.
answer	Script element that represents an answer to a question. Answers appear as data entry fields or as any of several types of UI elements, including check boxes, picklists, and multi-value groups.
translation	Text string used to display script elements in languages other than the original, so that scripts can be used in multilingual call centers. Screen appearance is determined by the type of script element (page, question, and so forth) to which the translation string is assigned. The maximum length of a translated string is 2000 characters.
branch	Transfers of control inside a SmartScript that define the display and processing sequence of pages or questions.
page section	Logical grouping of questions within a page which will display together to the user at one time.
Script Designer and Page Designer	Script Designer and Page Designer are visual tools that allow a script administrator to create scripts by graphically manipulating script elements as the script flow is being defined.
Script Sessions Table	Any script can have its questions and answers saved to a common answer table. This table is modeled with a parent table which displays the script name, position ID, contact ID, StartDate-Time, and campaign ID. The child table shows name-value pairs. This table is useful for later analysis of script sessions and answers.

SmartScript Screens

SmartScript provides administrative views, which you use to define and manage SmartScripts, and run-time user views, which display SmartScripts that have been set up for employees or customers.

Topics include:

- "The Employee's Screen"
- "The Customer Screen"
- "The SmartScript Administration Screens"

The Employee's Screen

The agent's interface to SmartScript is part of the standard Siebel Web client. The screen consists of a standard explorer view which allows the user to navigate through the script, and the SmartScript Player where the actual SmartScripts are displayed.

If a script is not launched by an incoming phone call by way of Siebel CTI, or called by Siebel VB or Siebel eScript, the agent must click the SmartScript screen tab to invoke SmartScript. When an agent opens SmartScript, a list appears from which the agent clicks on a hyperlink to select a script and a language. Only active scripts that have been translated into the selected language and are valid for the user's organization will be displayed.

Once a script has been opened, users can read or answer questions, move to the next or previous set of questions, cancel or finish a script, or finish it to be restarted later.

SmartScript Explorer

After the agent opens a script, a hierarchical view of the script, called the SmartScript Explorer, is displayed on the left side of the screen.

The SmartScript Explorer allows viewing and navigating through a script in a more dynamic, flexible way. It includes the standard Windows + and – icons that signify whether additional information can be displayed below the icon. Different icons represent different things:

- Green check mark icons signify questions that you have answered satisfactorily.
- Yellow question mark icons signify optional questions or prompts.
- Red question mark icons signify mandatory questions.

On the right side of the screen is the SmartScript Player in which questions and information text will be generated automatically based on the answers provided earlier in the script. Answer fields are represented by standard Siebel controls such as text boxes, pick applets, and drop-down list boxes.

By default, as the script is executed, the SmartScript Explorer displays a list of the questions on the current page as well as the status of those questions. As the agent records answers to questions and selects the next button, the SmartScript Explorer automatically reflects these developments.

Customer Dashboard

Displayed along the top of the agent's SmartScript screen is a text box called the SmartScript Customer Dashboard, which displays persistent data acquired from the script or from an outside source, such as Siebel CTI or a database query. The Customer Dashboard can, for example, display agent statistics such as average call time and call-queue status, as well as callers' answers to particularly important questions, such as those that elicit the customer's name or account number. This data is displayed no matter which page of a SmartScript is current. See Chapter 9, "Modifying the Dashboard" for overview information on updating the Customer Dashboard from previous releases of Siebel applications. See *Siebel Tools Reference* for more detailed instructions about the Customer Dashboard.

SmartScript Buttons

The following buttons appear in the SmartScript user interface of employee applications.

- Finish: Finishes the script session.
- Cancel: Cancels the script session.
- Finish Later: Allows the user to resume a SmartScript at the point where it was left off. Users can resume scripts from the My Saved Sessions view.

The Customer Screen

The SmartScript Customer screen differs in appearance from an employee's SmartScript screen mainly in that the explorer applet is not displayed. The customer screen allows a user to troubleshoot a problem or obtain information.

A user typically launches an SmartScript from a hyperlink on a Siebel customer application, from a custom button set up in the user interface, or from an email sent out by the Siebel eMarketing application. SmartScript questions and answers are displayed side-by-side.

Because the SmartScript user may be unfamiliar with script or scripting, the SmartScript screen does not display an explorer, a dashboard, or the Finish Later button.

The SmartScript Administration Screens

The SmartScript Administration screens contains multiple views in which a call center manager, business analyst, or system administrator can construct and maintain scripts.

SmartScript uses objects that are generically called *elements* to create the business process flow. The SmartScripts Administration screen contains views used to create and configure all the elements contained in a script, including questions, answers and translations. For more information on working with elements, see Chapter 2, “Fundamentals of SmartScript Creation.”

Fundamentals of SmartScript Creation 2

This chapter introduces all the elements that make up a SmartScript, and provides practical details to allow you to create, test, store, and release your own SmartScripts.

SmartScript Elements

Questions are the main element, or object, inside a SmartScript and include such attributes as answers and data indicating how a question should display. Questions are joined together with branches and groups of questions are contained inside pages. Scripts are the broadest element and contain groups of pages. Every aspect of each script element—from text properties, to branches, to events invoking related views in your Siebel application—can be modified without affecting other elements.

SmartScript also contains event handlers that support Siebel VB and Siebel eScript methods and interactions with outside processes. For more information, see Chapter 7, "Extending Scripts with Siebel VB and Siebel eScript."

Figure 1 displays each script element and its place within the hierarchy of elements.

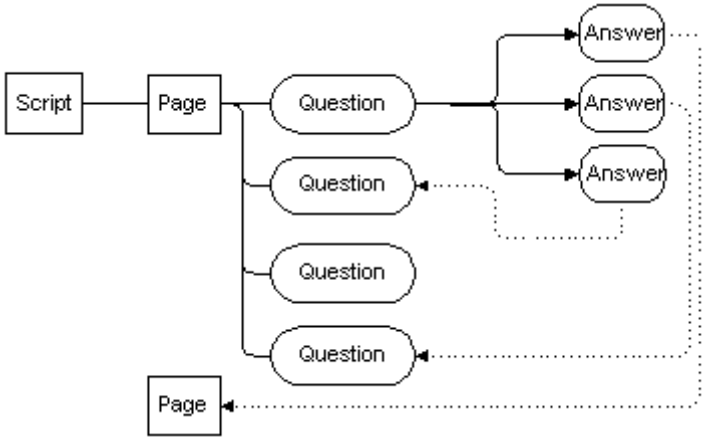

Figure 1. Scripts in the Hierarchy of Script Elements

Scripts

Scripts are the highest-order containers of script elements.

NOTE: Both conceptually and procedurally, the process of defining a script and adding pages to it closely resembles the process of creating a page and adding questions to that page.

Pages

Pages are groups of questions that are displayed together in a view when a script is executed. Pages should contain related questions, and questions should proceed in the order required by the workflow. Because pages are stored separately from the script itself, they can be used more than once in a script and can be used in multiple scripts.

Questions

There are two forms of question script elements: questions and information questions.

Questions elicit information from the caller or Web user. Questions also provide information about policies, options, and product descriptions.

Information questions are a different type of question. Information questions supply text to be read by the script user to guide the flow of a script or to supply information such as product descriptions, legal disclaimers, or step-by-step advice. There are no answers to store for information questions.

Answers

Answers are script elements that represent answers to questions. Answers appear as data entry fields or as any of several types of UI elements including check boxes, drop-down lists, or picklists. Questions and answers are linked by a one-to-many relationship.

Branches

Branches are transfers of control inside a SmartScript that determine the display sequence of pages or questions. All movement within a script must be explicitly defined using branches. There are two types of branches in SmartScript: *script branches* and *page branches*. Script branches transfer control from a question in one page to a question in another page, while page branches transfer control from one question to any other question within a single page.

NOTE: Page branches override script branches.

Most SmartScripts contain questions with multiple mutually exclusive answers. For example, a question may ask whether a customer is making a deposit to, rather than a withdrawal from, an IRA account. Those possibilities require different subsequent questions; therefore, you must create a separate branch for each possibility.

At other times, a SmartScript user may need to interrupt the expected flow of a script. For example, if a loan agent makes a call to offer a credit card, only to find that the target works for a competing bank, it makes no sense to continue with the standard script; the call should be ended as quickly, and as smoothly as possible by branching to the end of the script.

CAUTION: There are no built-in safeguards to prevent your creating branches that result in closed loops of questions. Therefore, you must test all your branches to be sure that they have an exit.

Data Storage in SmartScript

Answers gathered through SmartScripts may be stored in a number of ways. You also have the option of not storing questions at all. Storage methods are listed below:

- In a generic answer table: This table is always available without further configuration. This storage method is particularly useful when data is to be summarized later through an external system. Data stored in answer tables can be extracted for external processing by using the Siebel Enterprise Integration Manager. For more information, see *Siebel Enterprise Integration Manager Administration Guide*.
- In a particular business component field. For more information, see *Siebel Tools Reference*.

If the script Save Session parameter is set to Finished, Always, or Finished/Time-out, and the question save answer table flag is checked, the script session information including questions and answers can be viewed or analyzed in the Script Session Administration View.

Answers should be stored in business component fields when the SmartScript is collecting data that is represented in the system's business objects. For example, caller information collected during a script-controlled transaction should result in a new or updated record in the Contact business component.

Questions that accept answers, but do not store them, can be used to determine which branch should be followed. For example, a consumer information center might both supply information about rebate offers and answer questions about product safety. The question of which of these topics is the reason for a call may not be relevant to the particular call record, but the answer will probably determine which pages of the script should be displayed.

NOTE: The “Save Session” parameter and “Save Answer” flag may not be used independently of each other. They are designed to work together to save answers to the Sessions Tables. They are also independent of Save Buscomp, Save Field and Save Bus Object. There is no setting that affects when answers are saved to the Save BusComp fields.

To view saved SmartScripts in an employee application

1. From the employee application home page, click the SmartScript screen tab.
2. From the Show drop-down list, select My Saved Sessions.

To view saved SmartScripts as an administrator

1. Navigate to Site Map > SmartScript Administration.
2. From the Show drop-down list, select Script Sessions.

   A script that has been saved using the Finish Later button will be able to be restarted from this screen. If the contact ID is set during the script, it will populate into the script sessions table. In this case, a script session can be viewed by its association with a contact in the Contacts screen.

Best Practices For SmartScript Construction

The best practice for constructing SmartScripts is to create all your questions and answer elements first, and then create the structure of your script.

These steps fall into the order outlined below:

- Understand your business scenario and map out your design
- Create questions and translations
- Create answers
- Create pages
- Create the script
- Add questions to pages
- Add pages to the script

Script Design Tips

While the topic of detailed script design is outside the scope of this guide, the following general observations may prove helpful when you are creating scripts:

- The writing style should be conversational, that is, worded so as to seem part of a natural dialogue between the agent and an interested customer or prospect.
- A script should be flexible enough to let the agent respond to unforeseen questions, comments, and objections.
- A script should be general enough to address the needs of all potential customers or prospects, without forcing them into preexisting categories that may not match their needs.

Creating Questions

You create questions in the SmartScript Question Administration view. Because questions are stored separately from the script itself, they can be used more than once in a script and can be used in multiple scripts.

A question element can be an instruction to the agent or employee on how to proceed or to an agent on how to interact with the customer. Or, it can be a block of text to be read and acted on by the user.

See Chapter 3, "Working with Questions, Answers, and Translations," for more information on creating questions.

To create a question

1. Navigate to Site Map > SmartScript Administration > Questions.
2. Click New, and in the More Info form, enter the question's attributes.

Table 2 details the attributes a question can have. Table 3 on page 31 shows the question answer types you have to choose from.

Table 2. Question Attributes

Question Setting	Description
Name	Identifier used by the administrator for the question.
Answer Type	Data type for the answer to this question. See Table 3 for a list of answer types.
Answer Control	Indicates which answer control you will use to display your question, such as radio button, drop-down list and so on. For more information on answer controls, see "Displaying Answers Within a SmartScript" on page 55.
Pick Only	Indicates that the answer must be chosen from the list of answers attached to the question. If this box is not checked, you may still use a picklist, but agents may enter answers that are not on the list.
Must Answer	Indicates whether this question is optional, always required, or required *only* if it is reached. Note that when a question has branches from all of its answers, the Must Answer attribute must be set to Must Answer or Answer if Reached.
Default Answer	The default answer displayed for the question.
Save Answer Table	If the script session is being saved, selecting this flag will also save this question and answer to the answer table.

Table 3 describes the data types that are available as answer types for SmartScript questions.

Table 3. SmartScript Question Answer Types

Answer Type	Description
String	Alphanumeric characters
Integer	Whole number only
Number	Numerals only
Currency	Numerals only; uses currency code and exchange date for currency conversion
Boolean	Yes or No answer; usually displayed as a check box
Date	Date only
Date & Time	Date and time
Time	Time only

Translating Questions

You can translate questions displayed to a SmartScript user into any supported language. If you are going to translate questions into other languages, you must be sure to translate all questions within the script into that same language, as failure to do so may results in the script failing.

See Chapter 3, "Working with Questions, Answers, and Translations," for detailed information on creating question translations.

To create a question's translation

1. Navigate to Site Map > SmartScript Administration > Questions.
2. Click the Translations view tab, and click New.
3. Click in the Language row, and click the select button.
4. In the Pick Language Name pop-up window, select a language for your question, and click OK.

5 Click in the Question column, and enter the question text.

Repeat the above steps for every language in which the question will be displayed.

NOTE: Every question in the script must have a translation in the same language or languages in order to execute.

Creating Answers

You create answers in the SmartScript Questions Administration view.

It is possible to create selectable answers in the SmartScript definition itself, or to have the selection come from a business component. See Chapter 3, "Working with Questions, Answers, and Translations," for detailed information on creating answers.

To create an answer

1 Navigate to Site Map > SmartScript Administration > Questions.

2 In the Questions list, select the question for which you will be adding an answer.

3 Click the Answers view tab, and click New.

4 Enter a display order for this answer under this question.

Lower-numbered answers are displayed first. You may want to number the initial display order by tens to allow space for later additions and changes.

5 Enter the text or value for the answer, and its currency, if applicable.

The answer record is automatically saved as a child to the question. Repeat the above steps to add another answer to this question.

NOTE: Answers are saved to the business component only after the script completes. For more information see Chapter 3, "Working with Questions, Answers, and Translations."

Translating Answers

It is good practice to translate answers when the answer type for a question is a string, so that answers can be displayed in different languages. This step is not required for scripts only deployed in one language and optional for scripts deployed in multiple languages. If you do create answer translations, then when the script runs in a particular language, it will look for and use the answer translation as the answer label.

See Chapter 3, "Working with Questions, Answers, and Translations," for detailed information on creating answer translations.

To create a translation for an answer

1. Navigate to Site Map > SmartScript Administration > Questions.
2. In the Question list, select the answer you want to translate.
3. Click on the Translations view tab, and click New.
4. Click in the language column, and click the select button.
5. In the Pick New Language pop-up window, select a new language, and click OK.
6. Enter the text to use for this answer translation, and select Save Record from the view menu.

Creating Pages

You create pages in the SmartScript Page Administration view.

To create a page

1. Navigate to Site Map > SmartScript Administration > Pages.
2. Click New, and enter the name for the page in the Name field.

   The value you enter into the name field will aid in administration, in drop-down lists as an identifier when you build pages into scripts. This entry does not appear in the UI.
3. Click in the First Question field, and click the select button.

4 In the Pick Question pop-up window, select the first question that you want available in the page.

   Every page must have at least one question.

5 Click the Pages menu, and select Save Record.

Translating Pages

Page translations are titles shown in specified languages for the individual page tabs displayed in the SmartScript agent's view. Page translations must be specified for all languages in which the script will be run.

NOTE: Because the questions and answers receive their own translations, the only item to translate on a page is its title.

To create a translation for a page title

1 Navigate to Site Map > SmartScript Administration > Pages.

2 In the Pages list, select the page you want to translate.

3 Click the Translations view tab, and click the Translations menu, then select New Record.

4 In the Language field, click the select button.

5 In the Pick Language Name pop-up window, select the language, and click OK.

6 In the Label field, enter the text to use for this translation.

7 Click the Translations menu, and select Save Record.

   The translated page title is saved with the page of which it is a translated part.

Creating Scripts

Scripts are the highest-order elements that are used in SmartScript. They are assembled from pages containing questions. Questions, in turn, comprise question translations and answers. Branches between elements create the structure of the script, within which the elements are displayed. You can define scripts in the SmartScript Administration view.

To create a script definition

1. Navigate to Site Map > SmartScript Administration > Scripts.
2. Click New, and in the Name field, enter a name for the script.
3. Click in the Type field, and from the drop-down list, select the script type.

   For example, scripts used in eService must be type Instructions or Troubleshooting.
4. Click in the First Page field, and click the select button.

   The Pick Page pop-up window appears.
5. Use one of the following methods to select the first page to be displayed in the script:
   - Select a page from the list and click OK.
   - Click New to add a new first page, and an empty row will be added to the list.
6. Select a Save Session setting for the script.
   - **Always.** The session record and answers are saved whether the script is completed normally or canceled.
   - **Finished.** The session record and answers are saved only when the script is completed normally and the agent clicks Finish. This is the default setting.
   - **Finished/ Timeout.** The session record and answers are saved when the script is completed normally and the agent clicks Finished or when the session times out.

- **Never.** The session record and answers for the script are never saved.

NOTE: Only answers to questions that are marked Save Answer Table are saved.

7 Create translations for the script title, as described in "Translating Pages" on page 34.

8 Set the other values for the script. See the following table for parameter descriptions.

The script will be automatically saved.

Parameter	Description
Type	Reference field for the application of the SmartScript. Type is used to limit the SmartScripts that display in particular applications, in particular list applets, or are available for searching.
Jumping allowed	Checking this option allows users of a SmartScript to move in a non-sequential way through the script using the Previous button or using the SmartScript explorer.
BusComp	SmartScripts that will be invoked from the Script button on the Account, Contact, or Opportunity screens should have this field set to one of those business components. When the Script button is selected, the SmartScript which has the BusComp value equal to the business component of the current view will invoke automatically.
Description	An explanation for how the SmartScript will be used or its purpose. Used for building indices for searching in SmartScripts.
Estimated duration	Reference field for how long a script is expected to take to complete.
Duration In	Units for the estimated duration.
Organization	Organization in whose members should have access to the script.
OnCancel Gotoview	View to go to when the Cancel button is selected.
OnFinish Gotoview	View to go to when the Finish button is selected.

9 Attach other pages to the script. See "Adding Pages and Branches to Scripts" on page 41 for details.

Translating Scripts

Script translations are translations to different languages of the individual Script titles displayed in the Choose SmartScript dialog box, which appears when you first choose the agent's SmartScript screen.

NOTE: Because the questions, answers, and pages receive their own translations, the only item to translate in a script is its title.

To create a translation for a script title

1 Navigate to Site Map > SmartScript Administration > Scripts.

2 In the Scripts list, select the script whose title you want to translate from the Script list applet.

3 Click the Translations view tab, and click New.

4 Click in the Language field, and click the select button.

5 In the Pick Language Name pop-up window, select a language from the picklist, and click OK.

6 Click in the Label field, and enter the text you want to translate.

7 Click in the Dashboard text, and enter the text and parameters to be displayed in the Dashboard. See Chapter 9, "Modifying the Dashboard," for details.

8 Click the Translations menu, and select Save Record.

NOTE: The Customer Dashboard is only available in employee applications.

About the Script Designer and Page Designer

Use the Script Designer and Page Designer to create scripts using a graphical user interface that is similar to a flowchart designer.

The SmartScript Script Designer and the Page Designer screens are similar. Both screens feature a list at the top of the screen—the Script Designer displays a list of scripts, and the Page Designer displays a list of pages. At the bottom of the screen is the Designer view workspace. The palette provides page or question icons (depending on which designer view) and branch icons in both views which allow you to join the different elements. You drag script elements onto the workspace to create your scripts.

Once you drag an element into the workspace, you can move it to any location. Pages and questions are connected with branches by dragging the branch into the designer and making sure the ends of the branch connect to the connection points on the question or page node. A connection is made when the branch and node touch displays a large box. In the workspace, you can double-click on pages or questions that have branches to or from them. Double-clicking on a page in the Script Designer takes you to the Page Designer, where you can further define the page. Double-clicking on a question in the Page Designer takes you to the SmartScript Question Administration view, where you can further define the question. You can double-click on branch icons to change the question from which it is branching. (To change the question to which a branch is pointing, you must disconnect the branch and reconnect it.)

The four points displayed on each element icon on the workspace are called connection points. Use the connection points to attach branches to elements. From the shortcut menu (right-click) you can add points to a branch. You can drag the points on branches to reshape the branch. This feature is useful when two branches are overlaying each other in the workspace.

In the workspace, the shortcut menu (right-click) provides changes to the way the designer is viewed, such as zoom, and will persist until you navigate to another view. These options are described in Table 4.

Table 4. Script Designer and Page Designer Shortcut Menu

Shortcut Menu	Description
Edit	Allows you to perform the following edits on elements in the workspace: ■ Undo a move. ■ Redo a move. ■ Delete an element. ■ Add or remove points on a branch. ■ Move branch label text backward and forward along the branch line.
Layout	Allows you to perform the following edits on elements in the workspace: ■ Align multiple elements. ■ Make two elements the same size. ■ Move elements. ■ Expand elements.
Zoom	Zooms the workspace in or out by selected percentages.
Connection Points	Turns the connection points displayed on elements on and off.
Show Grid	Turns the grid on and off.
Snap to Grid	Aligns elements with the grid lines during a move.
Autosize	Extends the workspace after you drag an object to the extents of the workspace.

Adding Questions and Branches to Pages

Questions and branches can be added to pages to design the flow for the page. Each question or branch element can be added using the drag-and-drop UI of the Page Designer.

To add questions and branches to a page

1. Choose Site Map > SmartScript Administration > Pages.
2. In the Pages list, select a page to which you want to add questions.
3. Click the Designer view tab.

   NOTE: If you arrived at this view from the Script Designer, the page you double-clicked on is already selected in the Pages list applet.

4. Drag and drop the question icon from the Page Designer palette to the workspace.

   The Pick Question dialog box appears.

5. Select a question type from the list and click OK. Add the next question in the page designer using the same technique.
6. To connect these two questions with a branch, drag the branch icon from the Page Designer palette to the workspace and align the arrowless end of the branch with a connection point on the question from which you want to branch.

   If you select a question with multiple answers, SmartScript gives the option of selecting an answer for which the branch will be used, or a default branch. A default branch covers the case when a user's answer is not covered by another branch. If you choose not to create a default branch, SmartScript requires you to choose an answer to branch from.

7. Drag the arrow end of the branch to align it with a connection point on the question to which you want to branch.
8. Repeat the above steps to continue adding required branches and questions to the page.

To view the branches within a page

1. Navigate to Site Map > SmartScript Administration > Pages.
2. Click the Branches view tab, and select All Branches from the drop-down list.

   The Branches list appears, displaying all branches within the page.

Adding Pages and Branches to Scripts

Adding pages to scripts involves a nearly identical process to adding questions to pages.

To add pages and branches to a script

1. Navigate to Site Map > SmartScript Administration > Scripts.
2. Click the Designer view tab, and select the script into which you want to add or modify pages.
3. Drag the page icon from the Script Designer palette to the workspace.

   The Page Pick dialog box appears.
4. Select a page name, and click OK.

To add branches to a script

1. Drag the branch icon from the Script Designer palette to the workspace, and align the arrowless end of the branch with a connection point on the page from which you want to branch.

   The From Question Name dialog box appears.
2. Select a question from the list and click OK.
3. Drag the arrow end of the branch to align it with a connection point on the page to which you want to branch.

   The Next Question Name dialog box appears.
4. Select a question from the list and click OK.
5. Repeat the above steps to continue adding required branches and pages to the script.

To view the branches between pages

1. Navigate to Site Map > SmartScript Administration > Scripts.
2. Click the Branches view tab, and select All Branches from the Branches drop-down list.

3 The Branches list appears, displaying all branches between pages in the script.

Use this display to make sure that all the necessary branches have been added.

Releasing Scripts

Once you have completed and tested your script, you are ready to release it. Though it is optional to release a script, it is recommended to release it to improve its loading speed. Scripts should only be released when you are ready for production, otherwise you must rerelease scripts after each change made in order for the change to be visible when testing.

Releasing a script saves (or more properly, caches) a precompiled version of the SmartScript definition and all associated code (VB or JavaScript). Releasing saves the release-compiled script to a file on the file system. It also updates the SmartScript definition to indicate that a release file exists, and to create a pointer to the release file in its server location.

When a user invokes a released script, the SmartScript engine will look for a release copy of the script on the server. If the SmartScript engine does not find a release copy of the script on the server, it will copy a release copy of the script from the file system to the server and then execute the script from there. If a released version of the script does not exist on either the server or the file system, an error is raised. If a script is not released, SmartScript must compile and execute the script from the server, which causes the script to load slower.

When you release a script, you must select the translation of the script you want to release. If a script has multiple translations, you must release each translation individually.

The following is an overview of the release process and how it will work in your environment:

1. The released file is created and put on the Siebel File system under the FileSystem root directory; there is no subdirectory. This released file is saved as a .SAF file, which is compressed in the same manner as all other Siebel FileSystem files are compressed.
2. If a client starts the SmartScript, the compiled version is downloaded from the Siebel FileSystem to the server machine (into the [siebsrvrdir] \ServerDataSrc\files\sscript directory). The file is renamed as .ssc file.

3 If a client starts the SmartScript again, the file is read from server machine without accessing FileSystem.

CAUTION: If you change a released script, be sure to rerelease it. Otherwise, SmartScript will continue to use the previously released version of the script, which will not include the most recent changes.

To release a script

1 Navigate to Site Map > SmartScript Administration > Scripts.

2 In the Script list, select the script you want to release.

3 Click the Translations view tab, and select the translation you want to release.

4 Click the Scripts menu, and select Release.

After a script is released, a check mark appears in the Released field of the translation record.

NOTE: When you release a script, only the translation you selected is released.

Unreleasing Scripts

Unreleasing a script erases the pointer to the script file. The result is that each subsequent execution of the script is compiled from the database every time you run the script. Unreleased mode is the same as development mode.

To unrelease a script

1 Navigate to Site Map > SmartScript Administration > Scripts.

2 In the Script list, select the script to unrelease.

3 Click the Translations view tab, and select the translation to unrelease.

4 Click the Scripts menu, and select Unrelease.

The released version of the script is deleted.

NOTE: This does not delete the actual script, only the released or compiled version of the script.

Working with Questions, Answers, and Translations

3

This chapter explains how to create questions and answers in the Questions Administration view. Using the information contained in this chapter, you should be able to create SmartScript questions, answers, and translations and define where answers will be stored.

Creating Questions

The question is the basic element of a SmartScript, and is thus created first when you build a new SmartScript. You create questions in the Questions Administration view. Questions are stored separately from the script itself, and can be used more than once in a script as well as in multiple scripts.

Questions can serve various functions in your SmartScripts, such as:

- **Eliciting information.** Questions can be asked by a call center agent to elicit sales or service information from a customer. Over the Web, questions can be presented to a customer as a a survey or a series of questions to isolate a problem. The answers given may then be stored for later use.
- **Providing information.** A question can provide text to be read by an agent to a customer, or read directly by a customer on a Web site. Examples of this type of question are policy statements, legal disclaimers, and product descriptions. These types of questions do not have answers.
- **Guiding a process or question flow.** A single question can determine which path the script will follow. On the Web, a customer's answer to a single question dictates the use of one form or another, or one part of a form rather than another. The answer to a call center agent's question can lead that agent to a different series of questions.

Many of the fields in the Questions Administration form are related to storage of answer data (also known as question control data). The Answer Type and Must Answer fields are obvious examples. The fields labeled Save Business Object, Save Bus Camp, and Save Field all serve to define the location for answer data given in response to the question. The fields labeled Width, Height, Minimum and Maximum all refer to the user interface space provided for answers.

To create a question

1 Navigate to Site Map > SmartScript Administration > Questions.

2 Click New, and in the More Info form, enter the question's attributes.

Table 5 details attributes a question can have.

Table 5. Questions Form Field Descriptions

Question Setting	Description
Name	The question's name is a label that identifies the subject of the question. When you build a page, you select each question by name.
Answer Type	Data type for the answer to this question. ■ String – Alphanumeric characters ■ Integer – Whole number only ■ Number – Numerals only ■ Currency – Numerals only; uses currency code and exchange date for currency conversion ■ Boolean – Yes or No answer; usually displayed as a check box ■ Date – Date only ■ Date & Time – Date and time ■ Time – Time only
Answer Control	The type of answer control you would like to use. ■ Check box – single-select (Boolean) and multiselect check boxes ■ Default– Siebel 2000 and Siebel 7 logic for displaying answer controls ■ Dropdown– displays list of answer selections ■ None– no answer control; question text only ■ Radio Button– single select list of items displayed as radio buttons
Pick Only	Indicates that the answer must be chosen from the list of answers attached to the question.

Table 5. Questions Form Field Descriptions

Question Setting	Description
Minimum and Maximum values	Used with Date, Integer, Number, Date & Time, and Time answer types to constrain the values that can be entered. These fields can also be used with String answer types to indicate the minimum and maximum number of characters (bytes) allowed in the string. Note that in Asian (double-byte) languages, each character requires two bytes. Therefore, the minimum or maximum string length the user is allowed to enter in an Asian language will actually be half the number entered in these fields.
Auto Substitute Parameters	Specifies a list of parameters that can be used to automatically substitute values into the question text. In the question text, if brackets appear around a word, the word is interpreted as a substitution parameter. This parameter could be a user parameter set up in the SmartScript, a business component field, or a parameter pulled from an application, such as a CTI parameter. Any question text in brackets that is listed in the auto substitute parameters field will be converted into the current value for that parameter. Note: The [BC.Field Name] only works if the SmartScript is positioned on the relevant Business Component. If the user has positioned on a business component by scripting behind the SmartScript, nothing will be retrieved. Example: [User.Last Name], [User.First Name], [Contact.Phone Number]
Must Answer	Indicates whether this question is optional, always required, or required *only* if it is reached. Required only if reached means that if a user goes down a branch in the SmartScript where a question is displayed, the question must be answered before the user can proceed in the script. Always required is the strictest answer setting as the user cannot finish the script unless these designated questions are answered.
Default Answer	The default answer displayed for the question. This is one of the answers defined for the question. If the answer is to be selected from a pick applet, then the default answer will not be shown.

Table 5. Questions Form Field Descriptions

Question Setting	Description
Width	The width, in pixels, of the text box provided for answers.
Height	The height, in pixels, of the text box provided for answers.
Search Spec	Applies the entered search specification to make a specific record active in a business component. Search Spec can be used for many purposes. For example, a search spec can be used along with user parameters to search, for instance, in a particular BC for a record with some field value equal to an answer in a previous question (assuming the answer to the previous question had been saved to a user parameter). A search spec can also be used in conjunction with Auto Substitute Parameters to insert field values into the text of a SmartScript question. Search Spec syntaxes are the same as those used in Siebel Tools. For more information on setting up Search specifications see *Siebel Search Administration Guide*. For more information on Siebel Tools, see *Siebel Tools Reference*.
Save Business Object	The business object in which the answer to the question will be stored.
Save Bus Comp	The business component in which the answer to the question will be stored. Note that to save a question's answer to a business component field, the business object *and* the business component *and* the field name must be specified for the question. See "Displaying Answers Within a SmartScript" on page 55 for details.
Save Field	The field in the business object or business component table that is to contain the answer data, or that will be used to identify the specific record that will contain the answer data. Note that special steps may be required to set up a save field. See "Displaying Answers Within a SmartScript" on page 55 for details.

Table 5. Questions Form Field Descriptions

Question Setting	Description
Save User Parameters	Specifies a field in a BC as a user parameter. The record that is set as active in that BC supplies the value. (Note that a Search Spec can set the record as active.) The result is that the field value from the active record is saved to the parameter. One common usage is to insert the answer to one question into the text of a subsequent question. An answer can be comprised of a field value or values picked from a business component. The field variable can then be inserted into the text of another question using the Auto Substitute Parameters field. The result is that the variable value is inserted into the question text. If any user property variables are entered in the Save User Parameter field, then the answer values or any field from any BC record or both can be saved to this variable. For example, you can select a particular record in the question (such as a contact) and then save that record's ZIP Code to the user parameter, even though the selected answer was the last name, not the ZIP Code. If the answer includes multiple values from a picklist, then each of the fields accepted from the selected record in the picklist can be saved to multiple user parameters by separating the properties by commas. Syntax: If you want to save the answer to a question to a user parameter, enter the name of the user parameter. It is also possible to save the value of a field in the current record, such as (User Parameter Name, [BC.FieldName]. To save multiple fields, separate the user variables with a comma. For example, enter (User Parameter Name,[BC.field name]), (User Parameter Name2,[BC.field name]), (User Parameter Name3,[BC.field name])
Save Currency Field	The field in the business object or business component table that will contain the currency setting.

Table 5. Questions Form Field Descriptions

Question Setting	Description
Pick Applet	Indicates the pick applet that the end user will use to select and save business component record data as the answer data. When the end user clicks a select button, the pick applet opens as a dialog box, or as a drop-down combo-box. The end user can save a row of data to the specified business component. The Save Field in the business component must have a mapping to the picklist. For more information, see "Displaying Answers Within a SmartScript" on page 55.
Mvg Applet	Indicates the multi-value group (MVG) applet to be used to save answer data to the specified business component when that data includes multi-value fields. The MVG applet must be mapped to the specified Save Field. For example, to save Project Team data, you might select the Project Team list applet. For more information, see "Displaying Answers Within a SmartScript" on page 55.
Detail Applet	Indicates the Detail applet that will be used to save answer data of a specific configuration and format to the specified business component. The Detail applet must be mapped to the specified Save Field. For more information, see "Displaying Answers Within a SmartScript" on page 55.
Save Answer Table	Indicates whether the answer is to be saved to the generic answer table.
Currency	The currency code used to identify the saved currency data, if such data is saved.
Replication Level	Indicates different levels of replication for Siebel Remote: All, Regional, and None. The default state is All.

Defining Question Text and Translations

In SmartScript, the text of each question is treated as a translation, no matter what the language. Even if you only create questions in your native language, you must define the text of each question as a translation. For example, if you label a question in American English, you must then enter the question text, in American English, in the translation form.

NOTE: Once you have decided to translate into a particular language, you must translate all questions in the script to that language. If you do not translate every question within the SmartScript, your script may not run properly.

For information on the creation and translation of questions see Chapter 2, "Fundamentals of SmartScript Creation."

To create a question's translation

1. Navigate to Site Map > SmartScript Administration > Questions.
2. In the Questions list, select the question for which you want to define question text.
3. Click the Translations view tab, then click New.
4. In the Language field, click the select button.
5. In the Pick Language Name pop-up window, select the language in which you will create the question text and click OK.
6. Click the Question field, then type the text of the question in the chosen language.
7. From the Translations menu, select Save Record.

NOTE: Every question in the script must have a translation in the same language in order to execute.

For each additional language in which the question will be displayed, return to Step 3. Each translation is saved to the database with the question.

Displaying Answers Within a SmartScript

Each question within a SmartScript can be defined to contain answer controls, or be informational in nature. Informational questions do not require answer controls, and usually provide instructions, information or dialog to the user. You use answer controls when you want the user to provide responses to questions.

If you want to limit the possible answers a user may select from, you have the option of defining these answer options within the SmartScript definition as Question Answers, or reusing an existing pick or MVG applet already defined in Siebel Tools.

NOTE: SmartScripts cannot be configured to handle association applets.

In the simplest case, users select answers to a question from a simple drop-down list of answers that you define specifically for that question, using the Answer list menu options. These drop-down lists of answers are defined with respect to a unique question, and are actually an extension of the question definition.

To use a drop-down list, you chose the Pick Only option in the Question Form. To use single or multiselect pick applets, or detail applets, you would not check Pick Only.

To use radio buttons or multiselect check boxes, your answers must be set up as part of the SmartScript definition.

The following sections list Question Answer options.

Information Text

A question can be set up simply to display as informational or instructional text without any solicitation of a user answer to the question. This is useful for providing guidance to a user to answer subsequent questions or to display dynamically provided text using Text Substitution.

Text Box

A simple input control for users is a text box, where the user does not have the option to select from a fixed list of answer options, but instead can enter free text.

Drop-Down Lists

If the answers to a question are simple, single value answers, then you can create a simple drop-down list by defining each answer option in the Answers form in the script definition. You define a unique domain of answers for the question. Fields which are based on LOV's will also display the LOV values in a drop-down list based on the save field value.

Radio Buttons

You can display your answer choices as radio buttons in employee or customer applications. Radio buttons are similar to drop-down lists in that a only a single selection can be chosen by the user from a fixed list of answer options.

Multiselect Check Boxes

You can define answer options for a question, then have these options display as multiselect check boxes. You must include your answers as part of your SmartScript definition, then choose Check Box for your answer control in order to have your answer options display with heck boxes. If you select Boolean for your answer type, only one check box will appear.

Pick Applets

Any question can employ a pick applet for users to select answers to that question. In order to set up a question with a pick applet or MVG applet, it must be set to save the selected record to the appropriate field in the business component. Therefore, the save Business Object, save Bus Comp, and save field must all be entered. Branching cannot be defined for answers selected from pick applets. The pick applet or MVG applet selected must be set up in Siebel Tools to save to the Save Field in the entered business component.

NOTE: The listed applets that appear in each dialog box are not necessarily all valid. Valid applets must include a field that maps to the Save Field in the Save Buscomp. Selecting an appropriate applet requires some familiarity with Siebel Tools and the buscomps to which SmartScript must be linked. For guidelines on working with applets, see *Siebel Tools Reference*.

Detail Applets

Some applets have a detail applet defined for a control or list column. These are specialized applets that operate on that control or list column in a very specific way. Usually these are pop-up applets that allow end users to enter data that is configured or formatted for a specific purpose.

The detail applets that are most commonly used are the Currency Popup applet, and the File Attachment applet. If you use the Currency Popup applet, the agent sees an icon next to the question's input box at runtime. Clicking this icon causes the applet to pop up, allowing the agent to specify the currency when entering an amount. Without this detail applet, a calculator appears, and the currency cannot be changed from one answer to the next. In the case of a file attachment applet, the user will be able to add a file to the record.

NOTE: Branching from a question can only be defined for answers that are defined as part of the script definition and not those selected from pick applets or based on LOV's. In the case of branches defined on a question using multiselect check boxes, if multiple answers are selected by a user, the default branch will be used.

Answer Types and Answer Control Choices

Table 6 lists Answer types and the Answer Control choices you have for each type.

Table 6. Answer Types and Answer Control Choices

For This Answer Type...	Choose from One of These Answer Controls				
	Check box	**Default**	**Drop-Down**	**None**	**Radio Button**
String	Check box (if answer exists)	Drop-down (if Pick Only selected and answers exist) Text Box (if Pick Only not selected) Pick/MVG/Detail Applet (if defined for the question	Drop-down (if answers exist)	Question text only	Radio Button (if answers exist)
Integer	Box with icon for number applet	Box with icon for number applet	Box with icon for number applet	Question text only	Box with icon for number applet
Number	Box with icon for number applet	Box with icon for number applet	Box with icon for number applet	Question text only	Box with icon for number applet
Currency	Box with icon for number-currency applet	Box with icon for number-currency applet	Box with icon for number-currency applet	Question text only	Box with icon for number-currency applet
Boolean	Single select check box	Single select check box	Single select check box	Question text only	Single select check box
Date	Box with icon for Date applet	Box with icon for Date applet	Box with icon for Date applet	Question text only	Box with icon for Date applet
Date-Time	Box with icon for Date-Time applet	Box with icon for Date-Time applet	Box with icon for Date-Time applet	Question text only	Box with icon for Date-Time applet
Time	Box with icon for Time applet	Box with icon for Time applet	Box with icon for Time applet	Question text only	Box with icon for Time applet

Creating Answers

To define an answer for a question, use the Answers Administration form in the SmartScript Administration Questions screen.

If the question merely presents information, no answer is required. If the answer options are coming from a Siebel buscomp, then you do not need to enter answers into the script definition.

If the answer consists of words, you first define the answer in the base language. If you are required to provide a translation, you can then provide Answer Translations text in the different required languages, as needed for each answer. Answer translations are optional, but be aware, if you provide one translated answer within a script, you must translate every subsequent answer.

If an answer consists of numeric values (such as answers whose Answer Type is integer, number or currency) there is no need for translation, as long as the unit of measure is unmistakable.

For information on creating and translating answers see Chapter 2, "Fundamentals of SmartScript Creation."

To create an answer

1. Navigate to Site Map > SmartScript Administration > Questions.
2. In the Questions list, select the question for which you want to define answers.

   If Pick Only is checkmarked for the question, can have more than one answer, and as a result, you must specify all valid answers.
3. Click the Answers view tab, then click New.

4 Complete the fields, which are described in Table 7, and then click Save.

Table 7. Answer Forms

Answers Form Field	Entry
Number	Provide a number that will determine the display order for this answer under this question. Lower-numbered answers are displayed first. Recommendation: Assign numbers that are multiples of ten (10, 20, 30...) to allow space for later additions and changes.
Value	Define a name for this answer that is unique within the context of the question.
Currency	Optional: If the answer will include monetary amounts, click the select button to open the Pick Currency Code dialog box. Select the currency information appropriate for the country or region and click OK. The appropriate symbol for the currency will appear with this answer.

The new answer definition appears in the Answers list. The answer record is automatically saved as a child of the question. To add another answer to this question, repeat Step 3 on page 59 and Step 4.

If answer translations are needed, you may want to define a placeholder row for each translation in the Answer Translations list. Establish a policy and a procedure for providing translation text.

Translating Answers

When the Answer Type for a question is String and the Pick Only field is checkmarked, you may need to translate answers so that they can be displayed in the same language as the corresponding question translation.

NOTE: Answer translations are not required if you are deploying the SmartScript in one language. If no translations are used within a script, the language-independent answer value will be used for all translations. However, if you do translate answers within your script, you must translate every question with answers into the full set of languages into which the script is translated.

To create a translation for an answer

1. Navigate to Site Map > SmartScript Administration > Questions.
2. Click the Answers view tab, then in the Answers list, select the answer you want to translate.
3. Click the Translations view tab, then click New.
4. In the Language field, click the select button.
5. In the Pick Language Name dialog box, select the language for the answer translation text and click OK.
6. Click in the Answer field, type the text of the answer, and then click Save.

   This is the text that will actually display to the user when they run the SmartScript in that language.

Repeat this procedure for each additional language in which the answer must be displayed. Each answer translation is linked to the answer, which is in turn linked to a corresponding question.

Storing User-Provided Answers From Sessions

Answers provided by a user during a SmartScript session can be saved for reuse in the application or for analysis.

Saving to the Script Sessions Table

You have the option of saving header information about each script session and also specific questions and answers for the session. To do so, you must set the Script Save Sessions parameter and check the Question Save Answer Table flag for any question that should be logged as part of the Script Sessions information. This is useful when you do not require the answers to be saved to records and other business components for the rest of the application, but still want to save the answers provided during the script.

NOTE: The Save Session setting and Save Answer flag may not be used independently of each other. They are designed to work together to save answers to the Sessions Tables. They are also independent of Save Buscomp, Save Field, and Save BusObject.

The Call Script Runs and Call Script Run Answers components are based on the Script Sessions table and the Answers table, respectively. You can use these business components to store answers provided by the user during a Smart Script. They allow you to control whether a session is created at all and which questions will have their answers saved in the answers table. The Call Script Runs business component saves the script name, as well as date and time started and the employee name if the script is run from an employee application, or the contact name if the script is run from a customer application. In addition, the duration of the script is automatically saved as is the language in which the script was run. There is a one-to-many relationship between the Call Script Runs table and the Call Script Run Answers table. The answers table simply stores question and answer pairs. In other words, you may expect to get a single session record and multiple answer records associated with that sessions record for every script run.

Saving to a Business Component

Answers provided during the course a SmartScript session can be used to update existing records or create new records in any Siebel business component. You can specify the location to which SmartScript will save answer data by completing three fields in the Question Administration form: Save Business Object, Save Bus Comp, and Save Field.

SmartScripts can automatically create a new record in a business component and save the answers to fields in that business component using the Save Business Object, Save Bus Comp, and Save Field settings. Fields which are mapped from picklists should use pick applets. SmartScripts can be set to either update an existing record or create a new one, and the key logic occurs when a question is found which has the Save Bus Comp and Save Field defined. If there is already an active record set in that business component, then the SmartScript will assume that this is the record which should be updated. If there is not yet an active record in that business component, then a new record will be created and the answers saved to that record. The record commit in both cases occurs when the Finish button is selected.

If a pick applet is not used, then the answer provided for the question that is mapped to that Save Field will be saved to that field for the active record. If a pick applet is used, then it will behave the same way as if the pick applet is used in a standard Siebel view.

NOTE: Business component records are saved in the Siebel database only after the successful completion of a SmartScript. If the SmartScript fails or is canceled, the answers will not be committed to the business component.

Setting Up a Save Field for a Multi-Value Field

If the save field for an answer already has a picklist or multi-value field associated with it in the business component definition, you need not create new answers. However, you must specify the appropriate save field in the definition of your question.

For example, an employee wants to save a caller's address in the Street Address business component of the Accounts business object. If the Street Address business component were not part of the Accounts business object, you would first have to add it, or arrange to have it added, using Siebel Tools, to make it available. When setting up the question, you would enter Accounts in the Save BusObj field and Street Address (or any of the Address Multi Value Fields) in the Save Buscomp field.

To set up a save field for a multi-value field

1. Use Siebel Tools to make sure that the business component with the multi-value field you want to use is listed as a business object component for the parent business object.
2. If the business component you want to use is not listed, add the multi-value field as a business object component to the business object to which you want to save the data.
3. In the SmartScript Question Administration screen, click the Question Administration menu, and select New Record.
4. In the Save BusObj field, pick the parent business object from the picklist.
5. In the Save BusComp field, pick the business component with which the multi-value field is associated.
6. In the Save Field field, pick the multi-value field into which the data should be saved from the picklist.

Question Events Logic Sequence

Figure 2 illustrates the logical order and structure for how SmartScript uses VB code and search specs, creates new records, and does text substitution. For more information on using Siebel VB or eScript to extend your SmartScripts, see "Extending Scripts with Siebel VB and Siebel eScript" on page 95.

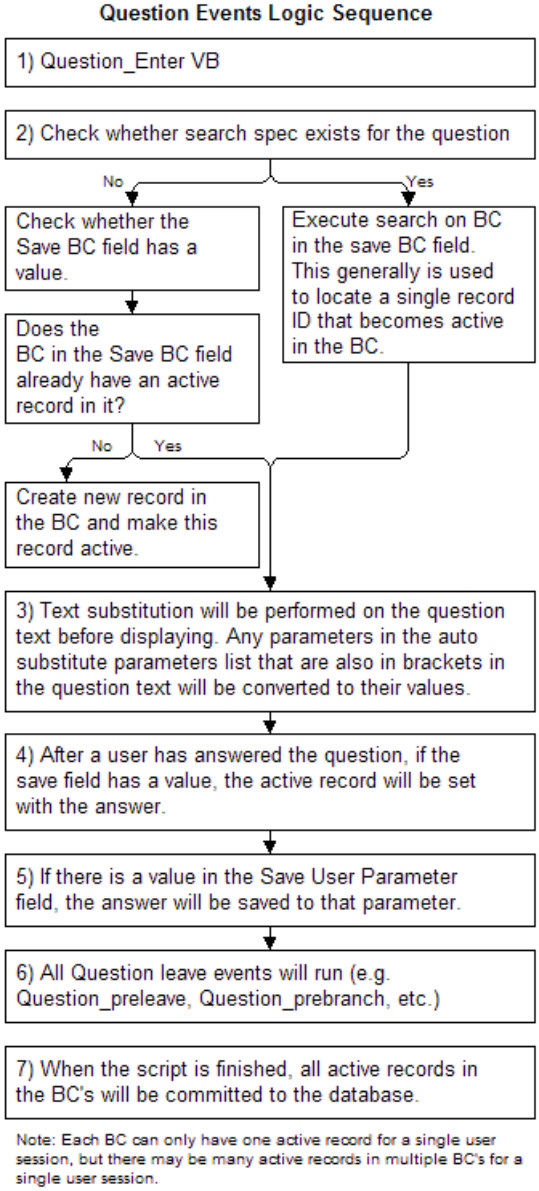

Figure 2. Question Events Logic Sequence

4 Upgrading SmartScripts

This chapter contains an overview of the tasks required to migrate a pre-Siebel 7 SmartScript to reflect current standards. Each task overview contains a cross-reference to detailed instructions within this book. This chapter also contains instructions on upgrading Script Wizards to SmartScripts.

Upgrading from a Previous Version of SmartScript

If you are upgrading from a previous version of SmartScript there are a number of performance and design implications having to do with migration of the data contained in your previously created SmartScripts to those you would like reflected in a post-Siebel 7 version of SmartScript. These steps are outlined in the following chapter.

NOTE: As a general rule, you should always un-release or re-release your released scripts after the upgrade process.

Overview of Migration from Pre-Siebel 7 Releases of SmartScript

The following section provides an overview of the issues involved with migrating data from releases previous to SmartScript 7. Cross-references point you toward other chapters in this book which provide instructions.

- **VB and eScripts.** Any VB or eScript that has been written inside SmartScript should be supported after the upgrade since it runs on the server. Any upgrade issues with upgrading the code are generic to the application and are discussed in the upgrade guide for the operating system you are using. For information on VB and eScripts see Chapter 7, "Extending Scripts with Siebel VB and Siebel eScript."

  NOTE: Browser scripts are not supported in SmartScripts. Support of specific VB and eScript code are subject to issues raised in the upgrade guide for your operating system.

- **Question displays.** Post-Siebel 7 SmartScript questions display differently than in previous releases. With version 7 and beyond, the maximum number of questions in a given page will display at once in a *page section*. Question Reveal functions are also not supported in post-Siebel 7 SmartScript (for example, show one at a time, all at once, and so on). You can still control what question displays by changing the location of branches in the script logic and the location of the VB or eScript. See Chapter 3, "Working with Questions, Answers, and Translations," for more information.

- **Layout of Questions.** The layout of SmartScripts is determined by the template. Administrators can modify the template if they want to change the layout. However, the ability to control the layout from the SmartScript definition is not supported in post-Siebel 7 SmartScript. See Chapter 5, "Customizing a SmartScript User Interface," for more information.
- **Text Formatting.** Any question, page, script, or answer text can be formatted using HTML tags. However, the Styles feature from previous releases is no longer supported. See Chapter 5, "Customizing a SmartScript User Interface," for more information.
- **Dashboard.** The SmartScript dashboard has been replaced with the customer dashboard which is available throughout the application. The active clock showing current time elapsed is only available for customers using SmartScript along with CTI. See Chapter 9, "Modifying the Dashboard," for more information.
- **Script Wizard.** Post-Siebel 7 SmartScript includes a feature to convert Script Wizard scripts to SmartScripts. Script Wizard will no longer be supported as a separate feature. See "Converting a Script Wizard into a SmartScript 7.5.3 Script" on page 70 for more information.

Converting a Script Wizard into a SmartScript

Siebel Script Wizard was a tool that created scripts that populated the fields of a single applet, instead of scripts that represented an entire transaction's workflow. Post-Siebel 7 SmartScript does not allow you to create or use Script Wizards, but it does help you convert them into regular scripts.

The conversion process involves the following steps:

- Use the convert utility to convert a Script Wizard into a SmartScript
- Establish the business object information for the script and the questions
- Select the correct picklist applet information for the reconstituted script
- Add a Save Currency Field if the field is a currency field

NOTE: Script Wizards appear only in the SmartScript Administration Script Wizard view.

To convert a Script Wizard into a SmartScript script

1. Navigate to Site Map > SmartScript Administration > Script Wizard.
2. In the Script Wizard Administration view, select the Script Wizard that you need to convert.

   You may import Script Wizards from a different environment, but all Script Wizards must be upgraded before they can run in the post-Siebel 7 environment.
3. Click the Script Wizard menu, and select Export.

   This is an optional step which allows you to make a backup of your script wizard before converting.
4. Click the Script Wizard menu, and select Convert.

   The confirmation view appears. Once you click OK, the script will no longer be available in the Script Wizard view, but only in the SmartScript view.

5 From the Show drop-down list, select Scripts, and in the Scripts form, check that the BusComp field in the script is the same as the business component on which the script button that invokes the SmartScript is based.

See "Creating Questions" on page 29 for more information.

6 In the Questions view, open the first question in the script, and complete the Save BusObj field.

See "Creating Questions" on page 48 for more information.

7 For each question, specify how answer data will be selected.

If answer data should be selected from an applet, the Save Field should specify the appropriate data for that applet, and a pick or MVG applet should be selected.

NOTE: If the answer field is a currency field you must add a Save Currency Field to make sure that the field is updated.

8 Check all other details in the script to make sure that nothing has been omitted and that it functions properly.

It is recommended that you verify and test the Script Wizard-based script as you would any new script.

9 Release the script.

NOTE: Entering the business component at the script level will allow the new SmartScript to be automatically executed from the script buttons on the Account, Opportunity, and Contact profile views.

Selecting Picklists for Questions

In earlier versions of Siebel applications, Script Wizard derived the pick applet and MVG applet automatically from the parent applet definition. Post-Siebel 7, you must explicitly specify this information.

Customizing a SmartScript User Interface 5

SmartScript provides a flexible user interface that allows easy modification. The answer controls that determine whether an answer appears as a text box, picklist, or pick applet are determined by values in the question definition. See Chapter 2, "Fundamentals of SmartScript Creation," for more information on working with questions.

Two other important ways in which the SmartScript user interface is designed are listed below.

- SmartScript's implicit page break rules, which determine whether certain questions will appear on a given page.
- SmartScript's HTML formatting capabilities, which allow you to determine how pages will be displayed to users, in terms of layout, typography, and other graphical elements.

Determining which questions should appear on the same page is a design decision that involves many factors. You have to consider a question's relationship to questions that precede and follow it, the need for branching logic, and many other requirements. In addition to the rules and design decisions you explicitly apply within a script to organize questions into pages, SmartScript itself applies rules that determine whether a question can appear on a given page. For example, if the definition of a question includes a certain type of VB event, the question may have to appear on the next page. To get the results you want, you must allow for and work with these implicit page break rules.

On another level, SmartScript allows you to apply and modify HTML formatting code to determine the way question and page data is displayed. You can work with HTML formatting on two levels, as follows:

- To enhance the typography or graphic display associated with an individual question, you use HTML code within the question Translation text.
- To change the design and layout of the page, you work with the HTML template.

Controlling the Questions Displayed on a Page

It is often convenient to place multiple questions on the same subject into a single page. If the resulting page is a serviceable unit, you can use and reuse this same page in different contexts, effectively reusing the same questions and question-controls to accomplish the same immediate tasks, but with a different higher-level goal.

SmartScript displays at once to the user all of the questions in a page section. A page section is a logical subset of all of the questions between page breaks. Page breaks are dynamically determined by SmartScript based on the following rules:

- A branch leads from a question to a new page or the end of the script.
- The last question on a page is completed.
- One of SmartScript's implicit page break rules causes a break.

About SmartScript's Implicit Page Break Rules

Sometimes a question cannot appear on the same page with a preceding or following question, regardless of other design criteria that you may be considering, because of the mechanisms involved in a question's definition. SmartScript processes each question, and if the processing of a question requires certain events, then SmartScript may have to end (or *break*) the current page to perform those events in a logical order.

For example, if a question has dynamic text substitution defined for it where the answer of the previous question determines the text for the question, then clearly there must be a page break between these questions. Or, a question may have code executed when the question is completed to check some values in the question answer before moving to the next question.

To determine where to break a page, SmartScript applies the following rules:

- If a question has a VB leave event such as Question_PreBranch, Question_PreLeave, or Question_Leave, SmartScript will always break a page after that question.
- If a question has a VB enter event such as Question_Enter or has a Search Spec, the page will always break before that question.

- If a question uses a User Parameter or a Buscomp Field to do Auto Substitution and a previous question in the same section sets that User Parameter or Buscomp Field (through Save Field), SmartScript will break a page before that question.
- If a question is on a Save BusComp and that buscomp is not positioned on any record, SmartScript will break the page before that question, unless the first question in that page is also on the same buscomp. This rule allows new records to be created.

SmartScript's implicit page break rules are based on the options used to define a question. As you work with these options, you should always consider their effects with respect to information flow and page design. It is often possible to put off a question that will cause a mandatory page break until you actually want the page to end. Or, it is often possible to move the location of VB or eScript code to a different location in the script in order to limit page breaks.

NOTE: You can manipulate SmartScript's implicit page break rules and exploit them to change the way questions are arranged into pages. For example, it is possible to add null or comment VB code that triggers a certain type of page break without including code that actually causes any other effect.

About HTML in the SmartScript User Interface

In SmartScript 7.5.3, all user interface (UI) elements are determined by HTML code and HTML templates. This means that an experienced HTML designer can redesign the appearance of a SmartScript or any element in it. Even a SmartScript Administrator with a rudimentary understanding of HTML can change the displayed appearance of a Question by adding basic HTML formatting code.

You can work with HTML formatting on two levels, as follows:

- To enhance the typography or graphic display associated with an individual question, you use HTML code within the question text—in a question's Translation field.
- To change the design and layout of the page, you work with the HTML template.

Formatting Question Text Using HTML Tags

Using standard HTML tags in the question translation text, you can format any text that is displayed to the user within a SmartScript. When a SmartScript page is rendered within the SmartScript Player applet, HTML tags also render in a standard manner. This means that it is a simple task to make any text bold, italicized, or a different size or color.

Example:

```
<B>Be sure to enter in a full description of your problem</B>
```

NOTE: Question labels are bold by default with the exception of Information questions. To change the default behavior of SmartScript, including bolding the text of Information questions, you must alter the Web Template: CCSmartScriptPlayerApplet.swt.

Adding Images

You can also add images to your SmartScript by putting image reference HTML tags into your question translations. You need to provide an explicit path to an image file like a GIF or JPEG and this image will display when that question translation text is rendered inside the SmartScript player applet.

Example:

```
<img src="\\goliath\images\explosion.gif">
```

Adding URLs

URLs can be added to questions to provide links to static, non-SWE-generated HTML pages. This is done by adding a URL reference tag inside the question translation text. These URLs should always be opened in a separate browser and should not reference Siebel pages.

Example:

```
You can go to the<a href="http://www.siebel.com"
target="_blank">Siebel Home Page</a> for more information
```

Formatting Example

You can use standard HTML in the question translation text to customize your scripts. The following text was copied directly from the Question field in the Translations form under the Questions view:

```
<b> Step 1:</b>Click the Add button to add a new data source. <BR>
<BR> <img src =
"\\main\demofile\demo38\my_images\SmartScriptImages\SmartScript_
buildX\DB2_Client_Connector_CreateDataSrc.gif">
```

Your server administrator can provide you with the correct locations for images, applets, stylesheets, and so on, if you wish to use such elements in a SmartScript.

Using HTML to Modify the Design Template

SmartScript pages are dynamically generated HTML content that combines the structure of the Web template with the UI elements and text from the SmartScript definition in the database. Most of the Web template is based on tables with SmartScript-specific SWE tags to substitute in values from the SmartScript definition.

Typically, you would change the template to make changes to the fundamental page design and layout. Much of the page design standard is based on tables, so making changes to table, row, and column parameters is relatively easy.

In Figure 3, questions are displayed from top to bottom in a single column. Each question starts directly below the preceding question and extends across the width of the single column. If you want to display a number of short fields side by side to make your page into a shorter form, you could modify the template to display data in two or more columns.

This template refers to a standard SmartScript page layout, the largest elements of which are represented in Figure 3. The template establishes the formatting for several dynamically generated tables.

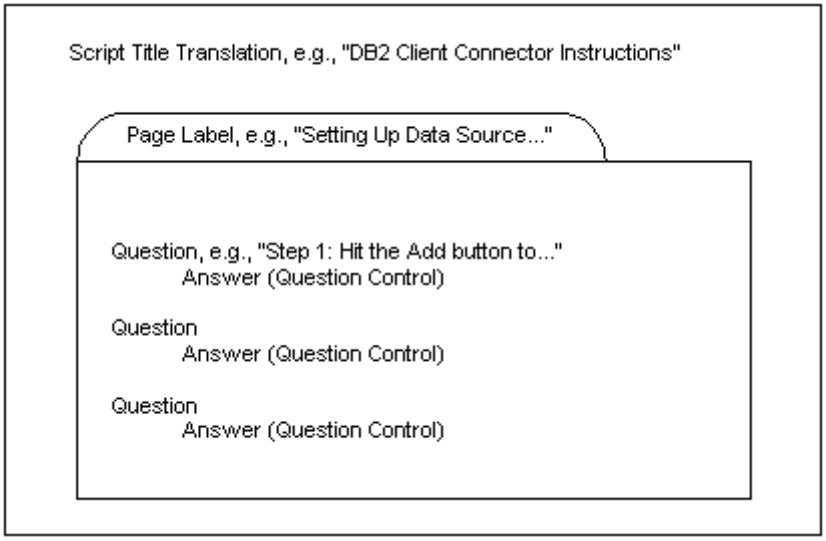

Figure 3. The Template Refers to This Basic Page Layout Design

NOTE: Question and question control (answer) pairs always use the same format determined in the Web template and repeat this format for all subsequent questions. This means that all question-answer pairs must keep the same format.

Using Siebel Tools to Modify the SmartScript View

There are some changes that can be made to the SmartScript view using Siebel Tools. The view on which the SmartScript player applet appears may be modified to add or remove other applets. For example, the tree control in the employee SmartScript views can be removed simply by removing the tree applet from the view.

NOTE: When you incorporate the SmartScript Player Applet (Player Only) into a custom view, set the Applet Mode property of the View Web Template Item to Edit. Otherwise, unexpected behavior may occur, such as data not displaying correctly.

Verifying, Testing, and Invoking SmartScripts 6

This chapter includes information on verification, testing and the different ways in which you can enable SmartScripts to be invoked.

The Verify Wizard

SmartScript includes a tool to test and verify scripts—the Verify Wizard. This tool can check for any or all of the following problems:

- Pages that cannot be reached through any branch
- Questions that cannot be reached through any branch
- Missing answer branches
- Missing translations

The Verify Wizard will also attempt to clean up dangling references and remove unreachable branches. These are problems which may occur when making deletions to an existing SmartScript. If you have these problems, they may cause a script to fail to execute or export but will be cleaned up with the Verify Wizard. You should not choose this option if you still are editing the script and plan on making revisions. Otherwise, Script elements with which you are still working may be deleted.

To invoke the Verify Wizard, click the Scripts List menu on the SmartScript Administration screen, and select Verify. The Wizard checks all the items that you select for verification. You must select Clean up dangling references and Remove all Unreachable Branches to execute these actions.

To verify a script

1. In the SmartScript Administration screen, click the menu button, and then select Verify.

   The Wizard starts and displays the Verify Intro view, in which you select the items to be verified.

2. Make your selections by clicking the check boxes, and then click Next.

   The Wizard checks the page branches and displays a list of any pages that cannot be reached.

3. Click next through the next three screens to see:

   - Pages which are not referenced properly in the script.

- Pages that are not reachable through branching.
- Other missing attributes.

4 Click Next.

The Wizard displays a summary of all errors found and indicates which were fixed automatically. Optionally, you can view a tree-structured display of the objects in the script.

5 Note any errors found so you can correct them, and then click Finish.

Be sure to verify your script again after you have made corrections, in case you have missed some errors or introduced new ones.

When your script has been verified with no errors, run it several times, using different answers at the various branch points to make sure that it behaves as you intended.

SmartScript Diagnostics

You can enable SmartScript diagnostics through event logging. To turn on SmartScript logging, set the event log level for GenericLog to 3 or 4.

The format is as follows:

For level 3:

```
SScript: [<Username>] <Session id> <TypeOfOperation(Start)>: <Name,
Id, Label, Lang, Curcy>

SScript: [<Username>] <Session id> <TypeOfOperation(Cancel, Finish,
Save, Next or Prev)>: <Current page, question and text>

SScript: [<Username>] <Session id> <TypeOfOperation(Cancel, Finish,
Save, Next or Prev)>: Error: <errorCode and errorMessage>

SScript:[<Username>] <Session id> <~CSSSWEFrameWebCAllScriptPlayer>
```

For level 4:

```
SScript: [<Username>] <Session id> <TypeOfOperation(Cancel, Finish,
Save, Next or Prev)>: <OperationStatus(OK)>
```

About Invoking SmartScripts

SmartScripts can be invoked in several ways:

- Through a Siebel application user interface, manually
- Through a Siebel application, automatically
- Through a reference from another SmartScript
- Through computer telephony integration (Siebel CTI)
- Through Siebel VB or Siebel eScript
- Through a link in an email or in a Web site

Invoking SmartScripts Through the User Interface

If a SmartScript is not opened by an incoming phone call with Siebel CTI, called by Siebel VB or Siebel eScript from inside a Siebel application, or opened by a hyperlink, an employee can start a SmartScript from the SmartScript user view.

To invoke a SmartScript from the standard SmartScript user view

1. In the Home page, click View > Site Map > SmartScripts.

   In some applications like Call Center, SmartScript appears as a screen tab.

2. Click the SmartScripts menu, and choose All SmartScripts to see all available SmartScripts.

3. Click a script hyperlink to open it.

4. To find a particular SmartScript, click the menu button and choose New Query.

   A form with one row appears.

5. Enter a search string in the Name field.

   For example, to find all SmartScripts whose names begin with the letters ABC, enter ABC*. Use the asterisk as a wild card.

6. Click the menu button again and choose Run Query.

   The SmartScript list reopens, listing all scripts with names that match the search string. You can also use predefined queries to view all SmartScripts for a particular language.

Setting SmartScripts to Open Automatically

SmartScripts can be set to run automatically when a user navigates to a particular view. To configure SmartScripts to run automatically, the SmartScript player applet must be added to the view, and a SmartScript must be associated with that view.

Use Siebel Tools to add the SmartScript player to a view. See *Siebel Tools Reference* for information. Use the SmartScript View Administration View to add SmartScripts to a view.

To set SmartScripts to open automatically

1. In the SmartScript Administration screen, choose Views from the Show drop-down list.
2. Select the view that will activate SmartScript.
3. Select the SmartScript to be run when this view is accessed.

When you have completed these steps, SmartScript can be activated by navigating to a view.

Setting Up SmartScripts in the Search Center

SmartScripts can be retrieved and started from a search performed in Siebel Search Center. The SmartScript available for retrieving can be limited by the SmartScript type, and you use the Script Description field to find keyword matches. For more information, see *Siebel Search Administration Guide*.

Invoking SmartScripts Through Siebel CTI

Siebel CTI can be configured to invoke a SmartScript on an agent's Siebel application. See *Siebel Communications Server Administration Guide* for details.

CTI parameters (from the telephone switch) can be processed and used within a SmartScript. These can be accessed through Siebel VB or Siebel eScript using the GetParameter function against the SmartScript object. Or, they can be used for text substitution just like any user parameter.

Invoking Scripts Through Siebel VB or Siebel eScript

SmartScripts are invoked programmatically, using the application invoke method RunSmartScript, or the applet invoke method RunCallScript. The InvokeMethod RunCallScript is used to invoke a SmartScript from an applet so the SmartScript will update the currently active record.

NOTE: The name of the SmartScript used in the 'RunSmartScript' and 'RunCallScript' application methods must be a basic (path) name of a SmartScript, which is not the translated SmartScript name which appears in the SmartScript player applet.

RunSmartScript

This function invokes a SmartScript programmatically.

Syntax *RunSmartScript callScriptName, pathId, language, currency, viewName*

Argument	Description
callScriptName	Name of the SmartScript, or “” if pathId is specified
AppletName	Name of the applet to be displayed if it is not the agent’s SmartScript applet
pathId	Row ID of the SmartScript, or “” if ScriptName is specified

Argument	Description
language	A language code
currency	A currency code
viewName	Name of the view to be displayed if it is not the agent's SmartScript view

Returns Not applicable

Usage For consistency, the arguments to RunSmartScript are identical to those of RunCallScript, except for the additional *viewName* parameter. This parameter allows a view other than the default agent's SmartScript view to be accessed. This is useful if an alternative view is configured for a specific purpose. The view specified must contain an instance of the SmartScript Player Applet to present the SmartScript interface to the agent.

Examples

Siebel VB and eScript Example

```
InvokeMethod "RunSmartScript", "Pentium II Script", "", "ENU", "USD
```

Siebel Applet Button Example

```
Sub ButtonScript_Click
TheApplication.InvokeMethod "RunSmartScript", "Pentium II Script
End Sub
```

OLE Example

```
Dim errCode as Integer
Dim Args(4) as String
Args(0) = "Pentium II Script"
Args(1) = ""
Args(2) = "ENU"
Args(3) = "USD"
Applet.InvokeMethod "RunSmartScript", Args, errCode
```

RunCallScript

This function is similar to RunSmartScript in that it programmatically invokes a SmartScript. However, it positions the script on the currently active record from the applet from which the script was invoked. This can be invoked by filling in the Method invoked property of the button to be RunCallScript. Alternatively, an administrator can invoke it through VB in order to pass in the additional parameters.

Syntax *RunCallScript SmartScript name, pathID, language, currency*

Argument	Description
SmartScriptName	Name of the SmartScript, or “” if pathId is specified
pathId	Row ID of the SmartScript, or “” if ScriptName is specified
language	A language code
currency	A currency code

Usage RunCallScript allows a SmartScript to be used to update an existing record and should be used behind a button on an applet. This works in an identical manner to the script button on the Account, Contact, and Opportunity Profile views.

Examples

Siebel Applet Button Examples

```
Function WebApplet_PreInvokeMethod (MethodName As String) As IntegerSub
       if MethodName = “MyRunCallScript” then
       InvokeMethod “RunCallScript”, “Pentium II Script”,””,”ENU”,”USD”
       WebApplet_PreInvokeMethod = Cancel Operation
       else
       WebApplet_PreInvokeMethod = ContinueOperation
       end if
End Function
```

Canceling, Finishing, and Resuming a SmartScript

Three InvokeMethod calls exist for canceling, finishing, and resuming SmartScripts. These methods are CancelSmartScript, FinishSmartScript, and Resume SmartScript.

NOTE: These methods operate on the SmartScript you have currently running.

CancelSmartScript and FinishSmartScript take no parameters.

The parameters for Resume SmartScript are listed below.

Resume SmartScript

Syntax *Resume SmartScript viewName, appletName*

Argument	Description
viewName	Name of the view to be displayed if it is not the agent's SmartScript view
appletName	Name of the applet to be displayed if it is not the agent's SmartScript applet

Invoking a SmartScript from a Currently Running SmartScript

A SmartScript can be invoked from another SmartScript using Siebel VB or Siebel eScript. The currently running SmartScript should be canceled or finished and then the RunSmartScript invoke method can be used to start a new SmartScript.

Usage You could create a SmartScript that has one SmartScript question start another SmartScript in the current language.

Example

Function Question_PreLeave () As Integer

```
Function Question_PreLeave () As Integer

   CurLang = GetCurrentValue
      Script.Finish
      If CurLang = "English" THEN
         TheApplication.InvokeMethod "RunSmartScript","HSN-New Customer",,"ENU",
```

```
    "USD"
else
    TheApplication.InvokeMethod "RunSmartScript","HSN-New Customer",,"ESP",
    “USD"
    END IF
    Question_PreLeave = ContinueOperation
End Function
```

Invoking SmartScripts Through a Web or Email Link

A SmartScript can be invoked from a URL link located in an email sent by the Siebel eCampaigns application or located on a Web page such Siebel eService. When a user clicks one of these URL links, a SmartScript is invoked.

For example, the eCampaigns application sends customers a marketing email with a URL link to an SmartScript called Buyer’s Profile, which collects information about the customer. Clicking on the Buyer’s Profile link invokes the SmartScript.

Another example would be in the eService application where SmartScripts can be presented as URL links accessible through the Troubleshoot Problems hyperlink, or as instructions, accessible through the View Instructions hyperlink.

Extending Scripts with Siebel VB and Siebel eScript

7

This chapter explains ways in which you can programatically extend the functionality of your SmartScripts though Siebel eScript and Siebel VB methods.

Improving Performance of Your Scripts

To improve performance of your SmartScripts you can work with a BusComp user property called "DeactivateBCField". Setting this to "TRUE" or "Y" will deactivate all BusComp fields except those used in SmartScript Questions. By default, this field is not deactivated, leaving all BusComp fields active.

To make this change in Siebel Tools, identify the SmartScript Player applet being used in the application and then change the user property for the business component on which this applet is based.

Activating Fields

If you are using VB or eScript to read records from a BusComp, you will need to programmatically call ActivateField when you are working with a field that is not used in your SmartScript Questions. It is recommended that you call ActivateField when you have changed the user property in the business component. If you have not changed the user property, you need not perform this task.

Siebel VB and Siebel eScript

Siebel VB (Visual Basic) and Siebel eScript (a JavaScript-like scripting language that shares tools with Siebel VB) can be used with all SmartScript elements. This chapter describes the Siebel VB and Siebel eScript methods that are specific to SmartScripts. The syntax for all of these methods is the same in both Siebel VB and Siebel eScript. Most of the examples provided in this chapter are in Siebel VB; however, the methods work the same way in Siebel eScript. For more information about Siebel VB and Siebel eScript, see *Siebel Tools Online Help*.

NOTE: Siebel eScript functions work in the same manner as standard JavaScript functions, and thus requires the trailing parenthesis () even when the function has no parameters.

Any function that can be performed using Siebel VB or Siebel eScript can be performed in the context of a SmartScript or a call script. In addition, SmartScripts are automatically configured with five Siebel VB and eScript procedures, which can be added to a question or script using the SmartScript Script Programs view and Question programs view. In addition, administrators can create their own custom procedures from these views. See Table 8 for a list of these procedures.

Table 8. Standard Siebel VB and Siebel eScript Events Used in SmartScripts

Name	Role
Script_Open	A post-event function called when the script is opened.
Script_Cancel	A post-event function called when the script is canceled.
Script_PreFinish	A pre-event function called when the user clicks the Finish button.
Script_Finish	A post-event function called after the script has been finished, but before data is saved, to allow any last-minute cleanup or post-processing.
Script_Save	An event function called before the normal script state has been saved to business components and the answer table and after the Script_Finish function has been run.

Each question element is configured with four such procedures. Other elements do not have Siebel VB or Siebel eScript procedures associated with them. These procedures can contain methods which can be used to further control the workflow of a script. See Table 9 for a list of these procedures.

Table 9. Standard Siebel VB and Siebel eScript Events Used in SmartScript Questions

Name	Role
Question_Enter	A post-event function called after the question has been entered and all pre-question processing is complete.
Question_PreLeave	A pre-event function called on the question before the user leaves it by jumping or proceeding (but not by Undo or Backup).
Question_Leave	A post-event function called after branching has been determined and all built-in validations have been performed.
Question_PreBranch	A pre-event function that allows the choice of answer to be replaced by the Siebel VB or Siebel eScript script.

SmartScript Object Types

Within a script, you can refer to Virtual Business Components just as you would refer to other Siebel Business Components. Virtual Business Components are a class of Business Components that access external data, which is data stored outside the Siebel database. Virtual Business Components allow you to present and manipulate external data through the Siebel user interface without having to replicate the data inside the Siebel data model.

Siebel VB and Siebel eScript recognize the following items as SmartScript object types:

- SmartScript
- SmartScriptPage
- SmartScriptQuestion
- SmartScriptAnswer

Methods Used with Scripts

The SmartScript object type represents the entire state of the SmartScript or SmartScript being executed. Only one script can execute at any one time.

To access the VB or eScript scripting area

1. Navigate to Site Map > SmartScript Administration > Scripts.
2. Click the Programs view tab.

   The scripting area appears.

To add script VB or eScript to a standard script event

1. Navigate to Site Map > SmartScript Administration.
2. Select the script and then click the Programs view tab.
3. Click in the Name field and select the event from the drop-down list.
4. Click in the Program Language field and chose the program language, then choose Save Record from the Programs menu.
5. In the Scripts applet below, enter your required code, then select Save Record from the Scripts menu.

NOTE: Each script has the option of using Siebel VB or Siebel eScript (JS). However, every script method for a given script has to use the same language.

To add a new event

1. Navigate to Site Map > SmartScript Administration.
2. Select the script, then click the Programs view tab.
3. Click New, then click in the Name field and select the type of event from the drop-down list.
4. Click in the Program Language field and chose the program language, then choose Save Record from the Programs menu.

5 In the Scripts applet below, enter your required code, then select Save Record from the Scripts menu.

Standard Methods for All SmartScripts

The methods in this section are preconfigured in SmartScript and are available by invoking a new event in the Programs view, then clicking in the Name field. You access the Programs view by clicking the Programs view tab on the SmartScript Administration screen.

Script_Open

Script_Open allows updating of the Dashboard. It also can be used to position the database on the appropriate record when saving answers to a BusComp.

Syntax Script_Open

Returns Not applicable

Usage Script_Open allows setting variables for use for complex branching, use in later questions, or in the dashboard. It can also be used to create a new record and populate some values using code.

Script_Cancel

Script_Cancel handles post-processing when a script is canceled.

Syntax Script_Cancel

Returns Not applicable

Usage This post-event is called to allow any last-minute cleanup or post-processing when the script is canceled. Script_Cancel will not be called if Cancel is called from the Script_Open event. If an error is raised during the Script_Cancel procedure, the error will be rolled back.

Script_PreFinish

Script_PreFinish is called when a script is finished.

Syntax Script_PreFinish

Returns An enumerated value indicating one of the script state indicators:

Value	Indicator	Description
0	ContinueOperation	Continue to Script_Finish
1	CancelOperation	Cancel operation
2	OperationComplete	Skip Script_Finish

Usage Script_PreFinish is called when the user requests (by clicking the Finish button) that the script be finished. This is a good place to check additional constraints on the script that were not set up through configuration. Script_PreFinish is declared as an integer.

Script_Finish

Script_Finish allows cleanup after a script is finished.

Syntax Script_Finish

Returns Not applicable

Usage Script_Finish is a post-event that is called to permit any last-minute cleanup or post-processing after the script has been finished. If an error is raised during the procedure, it is displayed to the user, but the script remains finished.

Script_Save

Script_Save can be used to save states not stored by normal means.

Syntax Script_Save

Returns Not applicable

Usage Script_Save is an event called before the normal script state has been saved to business components and the answer table, and after Script_Finish has been called. This is a good place to save additional states collected by the SmartScript and not stored by the normal mechanisms.

Other Preprogrammed SmartScript Methods

The script methods in this section can be used within any of the methods displayed in the drop-down list available from the SmartScript Programs Administration view.

Cancel

Cancel cancels the current script.

Syntax Cancel

Returns Not applicable

Usage The Cancel method stops the SmartScript's execution. This method only cancels the SmartScript's execution; it does not take you back to the original view or, as with the Cancel button, take you to the specified OnCancel view.

CurrentPage

CurrentPage returns the current page of the executing SmartScript.

Syntax CurrentPage

Returns The name of the current page of the executing SmartScript.

Usage CurrentPage is declared as a SmartScriptPage object.

CurrentQuestion

CurrentQuestion returns the current question of the executing SmartScript.

Syntax CurrentQuestion

Returns The current question of the executing SmartScript.

Usage The CurrentQuestion method returns the current question of the executing SmartScript. This will always return a question object if the script is actively executing. CurrentQuestion is declared as a SmartScriptQuestion object.

ExecutionState

ExecutionState returns the current state of a running SmartScript.

Syntax ExecutionState

Returns A representation of the current state of the SmartScript, as an integer value. The values are represented by the following constants, each of which is followed by its integer equivalent:

Script State	Integer Value
Not available	0
ssInitializing	1
ssRunning	2
ssFinished	3
ssCanceled	4

Usage ExecutionState returns 0 if the SmartScript object has not been set up for execution (used when listing available SmartScripts).

Finish

Finish causes the current script to be finished.

Syntax Finish

Returns Not applicable

Usage The Finish method causes the currently running script to be finished. The collected answers are saved as appropriate. This method can fail if the user has not answered all the questions that require answers (*must answer* questions).

GetCampaignId

GetCampaignId returns the campaign identification string.

Syntax GetCampaignId

Returns The campaign ID as a string.

Usage GetCampaignId returns the campaign ID as set up by launching a script from Siebel Campaigns, Siebel CTI, or the SetCampaignID method. This information, if available, is stored in the SmartScript session for calls.

Siebel VB Example

```
Dim CampaignId as String
CampaignId = Script.GetCampaignId
```

See Also GetCampContactId on page 106, GetContactId on page 107, and SetCampaignId on page 113.

GetCampContactId

GetCampContactId returns the campaign contact identification string.

Syntax GetCampContactId

Returns The campaign contact ID as a string.

Usage GetCampContactId returns the campaign contact ID for the campaign contact as set up by launching a script from Siebel Campaigns, Siebel CTI, or the SetCampContactID method. This information, if available, is stored in the SmartScript session for calls.

See Also GetCampaignId on page 106, GetContactId on page 107, and SetCampContactId on page 113.

GetContactId

GetContactId returns the contact identification string.

Syntax GetContactId

Returns The contact ID as a string.

Usage GetContactId returns the contact ID as set up by launching a script from Siebel Campaigns, Siebel CTI, or the SetContactID method. This information, if available, is stored in the SmartScript session for calls.

See Also GetCampaignId on page 106, GetCampContactId on page 106, and SetContactId on page 114.

GetLabelText

GetLabelText returns the translation of the current script name in the current language.

Syntax GetLabelText

Returns Language-specific label text for the script, as a string.

Usage GetLabelText is used when displaying script names, as in the Choose SmartScript dialog box.

GetPage

GetPage returns a page of the script by name.

Syntax GetPage(*name*)

Argument	Description
name	The name of a SmartScript page, as a string

Returns A page of the script, as a SmartScriptPage object.

Usage This method returns a page of the script when the page name is specified. This name is the non-translated name set during authoring, not the label of the page tab displayed for a particular language. GetPage is declared as a SmartScriptPage object.

Siebel VB Example

```
Dim IntroductionPage as SmartScriptPage

   Set IntroductionPage = GetPage("Introduction")
```

See Also CurrentPage on page 104 and Page on page 134.

GetParameter

GetParameter retrieves a value that has been assigned to a specified user parameter, a CTI switch parameter or a system parameter.

Syntax GetParameter(*ParameterName*)

Argument	Description
ParameterName	The name of a system parameter, as a string

Returns The value of the specified parameter, as a string.

Usage GetParameter can be used to retrieve a value assigned to a user parameter. The parameter name in this case must be prefixed with *User.* Usually a value is stored by using the SetUserParameter function.

GetParameter can be used to retrieve the current setting of a Siebel CTI switch parameter. The parameter name in this case must be prefixed with *CTI.* CTI switch parameters are set when SmartScript is invoked through Siebel Communications Server. For more information, see *Siebel Communications Server Administration Guide*.

This parameter can be used to retrieve the value of a system parameter. Table 10 lists the system parameters.

Table 10. System Parameters

System Parameter Name	Usage Definition
CampaignId	Set when SmartScript is invoked through Siebel Communications Server.
CampConId	Set when SmartScript is invoked through Siebel Communications Server.
ContactId	Set when SmartScript is invoked through Siebel Communications Server.
ScriptId	Script row ID.
ScriptName	Script name.

Table 10. System Parameters

System Parameter Name	Usage Definition
ScriptLabel	Script's translation (label).
Language Code/Language	Language in which the script is running.
Currency Code/Currency	Default currency.
System.RecordFound	Set to TRUE if there are records returned from the search spec defined on the question; set to FALSE otherwise.

Siebel VB Example

```
Dim ContactId as String
ContactId = GetParameter("ContactId")
SetUserParameter "ContactId", ContactId

GetParameter("CTI.ANI")
```

See Also

SetUserParameter on page 114.

GetQuestion

GetQuestion returns a question of the script by name.

Syntax GetQuestion(*name)*

Argument	Description
name	The name of a SmartScript question, as a string

Returns A question of the script, as a SmartScriptQuestion.

Usage This method returns a question of the script by name. This name is the non-translated name set during authoring, not the question text displayed for a particular language. GetQuestion is declared as a SmartScriptQuestion.

Siebel VB Example

```
Dim ContactQuestion as SmartScriptQuestion

   Set ContactQuestion = GetQuestion("First Name")
```

See Also GetQuestionEnable on page 130.

GetSessionId

GetSessionId returns a row ID of the session table where the script answers will be stored.

Syntax GetSessionId

Returns The row ID of the session table where the script answers will be stored according to the script's admin settings (Save Session column on the script and Save Answer Table flag on the script's questions). It returns a string. It will be empty if the script is not set to save session, or if GetSessionId is called from incorrect events (see below for the list of correct events).

Usage GetSessionId is declared as a string. This method should be called from either Script_Finish or Script_Save events.

Siebel VB Example

```
Dim ScriptSessionId as String
ScriptSessionId = GetSessionId
```

OriginalDashboardText

OriginalDashboardText returns the actual text in the dashboard field for the currently running translation.

Syntax

OriginalDashboardText

Returns

The text of the Dashboard as originally configured, as a string.

Usage

This method returns the configured value of the Dashboard text (Descriptive Text).

SetCampaignId

SetCampaignId sets the campaign ID, if gathered through script execution.

Syntax SetCampaignId(*ID*)

Argument	Description
ID	The ID of the sales or marketing campaign, as a string

Returns Not applicable

Usage SetCampaignId sets the campaign ID, if gathered through script execution. This information is stored in the SmartScript session.

See Also GetCampaignId on page 106, SetCampContactId on page 113, and SetContactId on page 114.

SetCampContactId

SetCampContactId sets the campaign and contact ID, if gathered through script execution.

Syntax SetCampContactId(*ID*)

Argument	Description
ID	The ID of the sales or marketing campaign and the contact, as a string

Returns Not applicable

Usage SetCampContactId sets the campaign and contact ID, if gathered through script execution. This information is stored in the SmartScript session.

See Also GetCampContactId on page 106 and SetContactId on page 114.

SetContactId

SetContactId sets the contact ID.

Syntax SetContactId(*ID)*

Argument	Description
ID	The ID of the contact, as a string

Returns Not applicable

Usage SetContactId sets the contact ID, if gathered through script execution. This information is stored in the SmartScript session.

See Also GetContactId on page 107, SetCampaignId on page 113, and SetCampContactId on page 113.

SetUserParameter

SetUserParameter assigns a value to a specified user parameter.

Syntax SetUserParameter *ParameterName, value*

Argument	Description
ParameterName	The name of a user parameter, as a string
value	The value to be assigned to *ParameterName*, as a string

Returns Not applicable

Usage SetUserParameter is used most commonly to store the value of a user parameter, which can be called in another subroutine by using GetUserParameter. It can also be used to assign a system parameter, obtained with GetParameter, to a user parameter. SetUserParameter is declared as a string.

Siebel VB Example

```
If ContactExists then
   SetUserParameter "ContactExists", "Y"
else
   SetUserParameter "ContactExists", "N"
ContactBC.NewRecord NewBefore
end if
```

See Also GetParameter on page 109.

StartPage

StartPage returns the configured name of the start page on the SmartScript itself.

Syntax StartPage

Returns The configured name—the original name before any translation—of the start page of the current SmartScript, as a SmartScriptPage object.

Usage StartPage is declared as a SmartScriptPage object.

StartQuestion

StartQuestion returns the configured name of the starting question on the SmartScript itself.

Syntax StartQuestion

Returns The configured name—the original name before any translation—of the first question of the current SmartScript, as a SmartScriptQuestion object.

Usage StartQuestion is declared as a SmartScriptQuestion object.

SubstituteText

Syntax SubstituteText*(text, "variable", "value")*

Argument	Description
text	A block of text, including the *variable* to be replaced.
variable	A string whose content is to be replaced by *value.*
value	The new text string to be inserted in the original block of text in place of *variable.*

Returns The original block of text with the new value substituted for the variable, as a string.

Usage This method substitutes a single string in the Text, found as [Variable] with the Value, and returns the changed text. In the *text variable, the variable and the value are enclosed in square brackets.*

NOTE: This method can be executed only once each time the method is called, because the entire string "[*text*]" is replaced with the *value* in place of the *variable.* However, the method can be called repeatedly to translate multiple values in one question or translation.

Siebel VB Example

The phrase “Are you calling from your car?” would be rendered as “Are you telephoning from your carriage?” if the function was configured to translate from US English to “Victorian” English as follows:

```
Dim VehCheck as String

   VehCheck = “Are you [phone] from your [vehicle]?”

      if (Victorian) then

      VehCheck = SubstituteText(VehCheck, “phone”, “telephoning”)
      VehCheck = SubstituteText(VehCheck, “vehicle”, “carriage”)

   else

      VehCheck = SubstituteText(VehCheck, “phone”, “calling”)
      VehCheck = SubstituteText(VehCheck, “vehicle”, “car”)

   end if
```

NOTE: The GetDashboardText and SetDashboardText methods will not affect the displayed values in the Customer Dashboard. See Chapter 9, “Modifying the Dashboard,” for details on how to affect the customer dashboard values.

Methods Used with Pages

The SmartScriptPage object type represents a single page of the SmartScript being executed. This object is accessible only programmatically.

GetHelpText

If any help text is present, GetHelpText returns the help text for the current page in the current language.

Syntax GetHelpText

Returns The language-specific help text associated with the page, as a string.

Usage Help text may be used as reference help text which can be captured and displayed to a user using VB within a question text.

GetLabelText

GetLabelText returns the translation of the current page name in the current language.

Syntax GetLabelText

Returns The translation of the name of the current page of the current SmartScript in the current language, as a string.

Usage GetLabelText returns the language-specific translation for the page name. This shows up in the page tab and can be used in error messages or other user interactions.

GetQuestion

GetQuestion returns the text of the specified question.

Syntax GetQuestion(*QuestionName*)

Argument	Description
QuestionName	The name of a question

Returns The question whose name is the parameter, as a string.

Usage GetQuestion is declared as a SmartScriptQuestion object.

GetQuestion returns the text of a question on the page when the question name is specified. This name is the non-translated name set during authoring, not the question text displayed for a particular language.

Script

Script returns the name of the script containing the page.

Syntax Script

Returns The name of the script containing the current page, as a SmartScript object.

Usage Script is declared as a SmartScript object.

StartQuestion

StartQuestion returns the starting question on the current page.

Syntax StartQuestion

Returns The first question of the current page.

Usage StartQuestion is declared as a SmartScriptQuestion object.

Methods Used with Questions

The SmartScriptQuestion object type represents a single question of the SmartScript being executed. This type of object is the one that is most often modified using Siebel VB or Siebel eScript.

To access the VB or eScript scripting area

1. Navigate to Site Map > SmartScript Administration > Scripts.
2. Click the Programs view tab.

   The scripting area appears.

To add script VB or eScript to a standard script event

1. Navigate to Site Map > SmartScript Administration.
2. Select the script, then click the Programs view tab.
3. Click New, then click in the Name field and select an event type from the drop-down list.
4. Click in the Program Language field and select the program language from the drop-down list, then choose Save Record from the Programs menu.
5. In the Scripts applet below, enter your required code, then select Save Record from the Scripts menu.

NOTE: Each question has the option of using Siebel VB or Siebel eScript (JS). Every question method has to use the same language per question. This means that the languages can be different for different questions, but within the same question, the languages must be the same.

Standard SmartScript Question Procedure

SmartScript questions are configured with the following procedures. They can be accessed by way of the drop-down list in the upper right corner of the right applet in the SmartScript Question Programs Administration view.

Question_Enter

Question_Enter is called when the processing of a question is complete.

Syntax Question_Enter

Returns Not applicable

Usage This post-event is called after the question has been entered and all pre-question processing is complete. This is a good place to change the question text or set the current value.

Question_PreLeave

Question_PreLeave is a pre-event called by certain methods before the user leaves a question.

Syntax Question_PreLeave

Returns Not applicable

Usage This pre-event is called on the question before the user leaves it by jumping or proceeding (but not by Undo or Backup). This allows question-specific validation to be performed.

Siebel VB Example

```
Function Question_PreLeave () As Integer

   Script.SetUserParameter "Current Product", GetCurrentValue
   Question_PreLeave = ContinueOperation

End Function
```

Question_PreBranch

Question_PreBranch lets a question be replaced, for purposes of choosing a branch, by the results of this function.

Syntax Question_PreBranch(*answer*)

Argument	Description
answer	The actual answer given to the question, as a string.

Returns An evaluation of an answer, represented by an integer.

Usage Question_PreBranch replaces the answer to a question by the results of this method. It is declared as an integer. The normal branching logic of matching answers to branches is performed, unless that event is overridden by this function. The final value of the Answer argument is compared against the answers given to determine which branch is to be taken out of this question. This allows programmatic processing to determine branching (among preconfigured branches), regardless of the actual stored answer. The value returned in the parameter will not be stored as the answer to the question, but will be used to choose the answer used for branching.

For example, if the caller supplies a bank account number, the function would evaluate the number to determine what type of account it is and would branch to questions for that type of account. The answer stored in the database, however, would be the actual account number given.

Question_Leave

Question_Leave is called after branching from the question.

Syntax Question_Leave

Returns Not applicable

Usage Question_Leave is called on the question after branching has been determined and all built-in validations have been performed.

Siebel VB Example

```
Sub Question_Leave
Dim ProductDesc As String
ProductDesc = Page.GetQuestion("Choose Product").GetCurrentValue
Script.SetUserParameter "ProductDesc", ProductDesc
End Sub
```

Other Preprogrammed SmartScript Question Methods

The question methods in this section can be used within any of the methods displayed in the drop-down list available from the SmartScript Question Programs Administration view.

AnswerType

AnswerType returns the data type of the answer.

Syntax AnswerType

Returns An integer representing a data type, as indicated in the following table.

Answer Type	Integer Value
ssString	1
ssInteger	2
ssNumber	3
ssCurrency	4
ssBoolean	5
ssDate	6
ssTrue	7
ssDateTime	8
ssInformation	9

Usage AnswerType returns the data type of the answer that the current question collects. Each data type is represented as an integer in the return value. AnswerType is declared as an integer.

CurrencyFieldName

CurrencyFieldName returns the name of the field in which the currency code is stored.

Syntax CurrencyFieldName

Returns The name of the field in which the currency code is stored, as a string.

Usage CurrencyFieldName returns the configured field name in which the currency code is stored. It is declared as a string. This method can be used only with questions that accept currency values (AnswerType = ssCurrency).

GetCurrentCurrencyCode

GetCurrentCurrencyCode returns the most recent currency code.

Syntax GetCurrentCurrencyCode

Returns The currency code entered in response to the current question, as a string.

Usage GetCurrentCurrencyCode returns the current currency code entered by the user for a question if it is a currency question. The currency code may have changed many times as the user worked through the script. GetCurrentCurrencyCode is declared as a string.

See Also GetInitialCurrencyCode on page 127 and GetPriorCurrencyCode on page 129.

GetCurrentExchangeDate

GetCurrentExchangeDate returns the exchange date for the most recent currency question.

Syntax GetCurrentExchangeDate

Returns The current currency exchange date entered in response to the current question, as a string.

Usage GetCurrentExchangeDate returns the current currency exchange date entered by the user for this question (if a currency question). This may have changed many times as the user worked through the script. GetCurrentExchangeDate is declared as a string.

See Also GetInitialExchangeDate on page 128 and GetPriorExchangeDate on page 129.

GetCurrentValue

GetCurrentValue returns the current answer to the current question.

Syntax GetCurrentValue

Returns The current value entered in response to the current question, as a string.

Usage GetCurrentValue returns the current value entered by the user for this question. This may have changed many times as the user worked through the script. GetCurrentValue is declared as a string.

Siebel VB Example

```
Sub Question_Leave

if GetCurrentValue() = "Y" then

   Script.Finish
   TheApplication.InvokeMethod "RunSmartScript", "Voicemail", "",
   "ENU", "USD"

end if
```

See Also GetInitialValue on page 128, GetPriorValue on page 130, and SetCurrentValue on page 136.

GetHelpText

If any help text is present for it, GetHelpText returns the help text for the current question in the current language.

Syntax GetHelpText

Returns The language-specific help text associated with the question, as a string.

Usage Help text may be used as reference help text which can be captured and displayed to a user using VB within a question text.

GetInitialCurrencyCode

GetInitialCurrencyCode returns the currency code for the question before the script was executed.

Syntax GetInitialCurrencyCode

Returns The initial currency code for the question, as a string.

Usage GetInitialCurrencyCode returns the currency code for the question (if it is a currency question) before the user started executing the script. This value is usually Empty unless the value came from a business component field or was set up by Siebel VB or Siebel eScript in a Script_Open procedure. GetInitialCurrencyCode is declared as a string.

See Also GetCurrentCurrencyCode on page 125 and GetPriorCurrencyCode on page 129.

GetInitialExchangeDate

GetInitialExchangeDate returns the exchange date of a currency question before the script was executed.

Syntax GetInitialExchangeDate

Returns The initial currency exchange date for a currency question, as a string.

Usage This function returns the initial currency exchange date for this question (if it is a currency question) before the user started executing the script. This value is usually Empty unless the value came from a business component field or was set up by Siebel VB or Siebel eScript in a Script_Open procedure. GetInitialExchangeDate is declared as a string.

See Also GetCurrentExchangeDate on page 126 and GetPriorExchangeDate on page 129.

GetInitialValue

GetInitialValue returns the value of an answer before the script was executed.

Syntax GetInitialValue

Returns The initial value for the question, as a string.

Usage GetIntialValue returns the value for the question before the user started executing the script. This value is usually empty unless the value came from a business component field or was set up as a Default Answer in the SmartScript Question Administration view or by Siebel VB or Siebel eScript in a Script_Open procedure. It is declared as a string.

See Also GetCurrentValue on page 126 and GetPriorValue on page 130.

GetPriorCurrencyCode

GetPriorCurrencyCode returns the currency code entered the previous time the question was reached.

Syntax GetPriorCurrencyCode

Returns The previous currency code for the question, as a string.

Usage This method returns the currency code for the question (if it is a currency question) before the user reached it most recently. This will be either the same value as returned by GetInitialCurrencyCode, if the user has never entered the question, or the value as returned by GetCurrentCurrencyCode, after the user left it the last time. The function is usually used only for the current question. GetPriorCurrencyCode is declared as a string.

See Also GetCurrentCurrencyCode on page 125 and GetInitialCurrencyCode on page 127.

GetPriorExchangeDate

GetPriorExchangeDate returns the exchange date code used the previous time a question was reached.

Syntax GetPriorExchangeDate

Returns The previous exchange date code for the question, as a string.

Usage GetPriorExchangeDate returns the currency exchange date for this question (if a currency question) before the user reached it most recently. This will be either the same value as returned by GetInitialExchangeDate, if the user has never entered the question, or the value as returned by GetCurrentExchangeDate, after the user left it the last time. This function is usually used only for the current question. GetPriorExchangeDate is declared as a string.

See Also GetCurrentExchangeDate on page 126 and GetInitialExchangeDate on page 128.

GetPriorValue

If the user has reached the question more than once, GetPriorValue returns the previous answer.

Syntax GetPriorValue

Returns The prior value of the question, as a string

Usage GetPriorValue returns the value for this question before the user reached it most recently. This will be either the value returned by GetInitialValue, if the user has never entered the question, or the value returned by GetCurrentValue, after the user left it the last time. This function is usually used only for the current question. GetPriorValue is declared as a string.

See Also GetCurrentValue on page 126 and GetInitialValue on page 128.

GetQuestionEnable

The GetQuestionEnable method returns the enable state of the current question.

Syntax GetQuestionEnable

Returns TRUE if the question is enabled and FALSE if the question field is disabled.

Usage The GetQuestionEnable method returns the enable state of the current question. When the question is enabled, the user can modify the question's answer. If the question is disabled, the question's answer is read-only.

Siebel VB Example

```
if GetQuestionEnable = TRUE then
   Script.SetUserParameter “Order_Product_Enabled”, “Yes”
end if
```

GetQuestionText

GetQuestionText returns the text of a question.

Syntax GetQuestionText

Returns The displayed text of the question, as a string.

Usage GetQuestionText returns the displayed question text. (The original configured text can be retrieved with OriginalQuestionText.) It is declared as a string.

Siebel VB Example

```
Sub Question_Enter
Dim QuestionText as String
   QuestionText = GetQuestionText
End Sub
```

See Also SubstituteText on page 116, OriginalQuestionText on page 133, and SetQuestionEnable on page 137.

GetSaveBusComp

GetSaveBusComp returns the business component configured to store the answer.

Syntax GetSaveBusComp

Returns The instance of the business component in which the answer is to be stored, as a BusComp.

Usage GetSaveBusComp returns the instance of the business component used to store the answer in. If no field and business component are configured on this question, it returns nothing. GetSaveBusComp is declared as type BusComp.

Siebel VB Example

```
Dim BC as BusComp
Dim Q as SmartScriptQuestion

Set Q = Page.GetQuestion("Intake Interview: Patient Name")

Set BC = Q.GetSaveBusComp

BC.NewRecord NewBefore
```

See Also SaveBusCompName on page 134.

GetSaveBusObj

The GetSaveBusObj method returns the business object configured to store the answer.

Syntax GetSaveBusObj

Returns The instance of the business object in which the answer is to be stored, as a BusObj.

Usage This function returns the instance of the business object used to store the answer. If no business component or business object is configured on this question, it returns a null set. GetSaveBusObj is declared as type BusObj.

See Also SaveBusObjName on page 135.

HasDefaultAnswer

HasDefaultAnswer returns a Boolean value indicating whether or not the question has a default answer.

Syntax HasDefaultAnswer

Returns TRUE if the question is configured with a default answer, FALSE if not.

Usage HasDefaultAnswer is declared as Boolean.

MustAnswer

MustAnswer returns a value indicating whether or not the question is required.

Syntax MustAnswer

Returns One of the integer values indicated in the following table.

Return Value	Description
0	The question is optional.
1	The question is required.
2	The question is required when reached.

Usage This function returns one of three values indicating whether the question is set to optional, required, or answer if reached.

OriginalQuestionText

OriginalQuestionText returns the original text of the question.

Syntax OriginalQuestionText

Returns The configured value of the question text, as a string.

Usage This function returns the configured value of the question text. (The actual value displayed is obtained through GetQuestionText.)

Siebel VB Example

```
Dim Text as String
Dim Orig as String
Dim Lname as String

   Orig = Question.OriginalQuestionText
   Lname = Script.GetParameter ("PersonName")

Text = SubstituteText (Orig, "PersonName", Lname)

SetQuestionText Text
```

See Also GetQuestionEnable on page 130 and SetQuestionEnable on page 137.

Page

Page returns the name of the current page.

Syntax Page

Returns The name of the page containing the question, as a SmartScriptPage.

Usage Page is declared as type SmartScriptPage.

See Also GetPage on page 108.

Script

Script returns the name of the script containing the question. It is declared as type SmartScript.

Syntax Script

Returns The name of the script containing the current question, as a SmartScript.

Usage Script is declared as a SmartScript.

SaveBusCompName

SaveBusCompName returns the name of the business component that stores the answer.

Syntax SaveBusCompName

Returns The name of the business component in which the answer is to be stored, as a string.

Usage This method returns the configured business component in which the answer is stored. This must be specified if a field is specified through SaveFieldName.

See Also GetSaveBusComp on page 131, SaveBusObjName on page 135, and SaveFieldName on page 135.

SaveBusObjName

SaveBusObjName returns the name of the business object that stores the answer.

Syntax SaveBusObjName

Returns The name of the business object in which the answer is to be stored, as a string.

Usage This function returns the configured business object in which the answer is stored. This must be specified if a business component and field are specified through SaveBusCompName and SaveFieldName, respectively. It is declared as a string.

See Also GetSaveBusObj on page 132, SaveBusCompName on page 134, and SaveFieldName on page 135.

SaveFieldName

SaveFieldName returns the name of the field that stores the answer.

Syntax SaveFieldName

Returns The name of the field in which the answer is to be stored, as a string.

Usage SaveFieldName returns the configured field name in which the answer value is stored. This may not be present for all questions, as not all questions store their answers in a field. (Some questions do not save answers at all.) It is declared as a string.

See Also SaveBusCompName on page 134 and SaveBusObjName on page 135.

SetCurrentValue

SetCurrentValue is a procedure to set the value for the answer to a question.

Syntax SetCurrentValue(*value*, [*CurrencyCode, ExchangeDate*])

Argument	Description
value	The answer value to be given for the current question.
CurrencyCode	The currency code for the currency in which the answer is to be expressed, if the question is a currency question.
ExchangeDate	The exchange date for the currency, if the question is a currency question.

Returns Not applicable

Usage The SetCurrentValue procedure sets the value for the question as though the user had entered it. All validation and branching is activated. If the question is a currency question (that is, the answer type is “Currency”), the currency code and exchange date should also be specified in the function’s parameters. The currency code and exchange date parameters are optional.

If a question is to be saved to a business component field, SetCurrentValue() should not be used for this question until after the SmartScript has been correctly positioned on that business component.

SetCurrentValue is declared as a string.

Siebel eScript Example

```
function Question_Enter (){
SetCurrentValue(“Hello”);
}
```

See Also GetCurrentValue on page 126.

SetQuestionEnable

SetQuestionEnable sets the enable state of the current question.

Syntax SetQuestionEnable(*Enabled*)

Argument	Description
Enabled	A Boolean value that enables (TRUE) or disables (FALSE) a question.

Returns Not applicable.

Usage The SetQuestionEnable method sets the enable state of the current question. When the question is enabled, the user can modify the question's answer. If the question is disabled, the question's answer is read-only.

Siebel VB Example

```
SetQuestionEnable (FALSE)
```

Siebel eScript Example

```
SetQuestionEnable (false);
```

SetQuestionText

SetQuestionText changes the displayed text for a question.

Syntax SetQuestionText(*text*)

Argument	Description
text	The new text to be substituted

Returns Not applicable

Usage This procedure changes the displayed question text. Note that no automatic substitutions in the question text are supported.

Siebel VB Example

```
Dim City as String

if City = "" then

   QuestionText = SubstituteText(QuestionText,"City", "[No City
   specified]")

else

   QuestionText = SubstituteText(QuestionText,"City", City)
   Script.SetUserParameter "City", City

end if

SetQuestionText(QuestionText)
```

See Also GetQuestionEnable on page 130 and SubstituteText on page 139.

SubstituteText

See "Other Preprogrammed SmartScript Methods" for a description and a VB example of this method.

See Also GetQuestionEnable on page 130 and SetQuestionEnable on page 137.

WasAnswered

WasAnswered returns a Boolean value indicating whether the question was answered.

Syntax WasAnswered

Returns TRUE if the user answered the question; FALSE if not.

Usage WasAnswered returns TRUE if the user answered this question or accepted a default answer and FALSE otherwise. The exceptions to this are Information questions and questions with an answer type of Boolean with no default answer; these questions will return TRUE even if the agent merely passes through them.

WasAnswered is declared as type Boolean.

Sample Siebel VB and Siebel eScript Methods

The first three Siebel VB and Siebel eScript code samples are designed to work as a unit, but are separated here for ease of understanding.

Dynamic Questions

The following sample code implements dynamic insertion of an answer into a question text. Specifically, it inserts the caller's title and last name, as determined by previous questions, into the target question. The question might be entered into SmartScript as follows:

Hello. Is this [Contact.M/M] [Contact.LastName]?

Siebel VB Code

```
Sub Question_Enter

   Dim MM as String
   Dim LastName as String
   Dim Q as SmartScriptQuestion
   Dim Text as String

   Set Q = Page.GetQuestion (“Contact: Mr/Ms”)
   MM = Q.GetCurrentValue

   Set Q = Page.GetQuestion (“Contact: Last Name”)
   LastName = Q.GetCurrentValue

   Text = OriginalQuestionText
   Text = SubstituteText (Text, “Contact.M/M”, MM)
   Text = SubstituteText (Text, “Contact.Last Name”, LastName)
   SetQuestionText (Text)

End Sub
```

Siebel eScript Code

```
function Question_Enter ()
{

   var MM;
   var LastName;
   var Q;
   var Text;

   Q = Page.GetQuestion ("Contact: Mr/Ms");
   MM = Q.GetCurrentValue ();

   Q = Page.GetQuestion ("Contact: Last Name");
   LastName = Q.GetCurrentValue ();

   Text = OriginalQuestionText ();
   Text = SubstituteText (Text, "Contact.M/M", MM);
   Text = SubstituteText (Text, "Contact.Last Name", LastName);
   SetQuestionText (Text);

}
```

Finding a Contact

The following Siebel VB and Siebel eScript code samples look for the current caller in the database.

Siebel VB Code

```
Sub Script_Open

   Dim FirstQuestion as SmartScriptQuestion
   Dim ContactBC as BusComp
   Dim ContactId as String
   Dim ContactExists as Integer

      Set FirstQuestion = StartQuestion
      Set ContactBC = FirstQuestion.GetSaveBusComp

      ContactBC.ActivateField "Id"
      ContactId = GetParameter("ContactId")

      If ContactId <> "" then
```

```
         ContactBC.SetViewMode 3
         ContactBC.SetSearchSpec "Id", ContactId
         ContactBC.ExecuteQuery ForwardBackward
         ContactExists = ContactBC.FirstRecord()

         If not ContactExists then
              ContactBC.NewRecord NewBefore
         end if

      else

         ContactBC.NewRecord NewBefore

      end if

End Sub
```

Siebel eScript Code

```
function Script_Open ()
{

   var FirstQuestion;
   var ContactBC;
   var ContactId;
   var ContactExists;

   FirstQuestion = StartQuestion ();
   ContactBC = FirstQuestion.GetSaveBusComp ();

   ContactBC.ActivateField ("Id");
   ContactId = GetParameter ("ContactId");

   if (ContactId != "")
   {

      ContactBC.SetViewMode (3);
      ContactBC.SetSearchSpec ("Id", ContactId);
      ContactBC.ExecuteQuery (ForwardBackward);
      ContactExists = ContactBC.FirstRecord ();

         if (!ContactExists)

         {
            ContactBC.NewRecord (NewBefore);
         }
```

```
   }
   else
   {

      ContactBC.NewRecord (NewBefore);

   }

}
```

Complex Branching

The following sample code demonstrates complex branching, setting branching activity (and the answer to a question) based on whether a contact existed or a new one was created. The code relies on the preceding sample for data acquisition and database querying.

Siebel VB Code

```
Function Question_PreBranch (Answer As String) As Integer

   Dim ContactBC as BusComp
   Dim ContactExists as Integer

      ContactBC = GetSaveBusComp
      ContactExists = ContactBC.FirstRecord()

   if ContactExists then

      Answer = “Y”

   else

      Answer = “N”

   end if

   Question_PreBranch = ContinueOperation

End Sub
```

Siebel eScript Code

```
function Question_PreBranch (&Answer)

{
```

```
var ContactBC;

var ContactExists;

ContactBC = GetSaveBusComp ();

ContactExists = ContactBC.FirstRecord ();

if (ContactExists)
{

   Answer = "Y";

}

else

{

Answer = "N";

} return (ContinueOperation);
```

Siebel VB Code

```
Function Question_PreLeave() As Integer

   Script.Finish

   TheApplication.InvokeMethod “RunSmartScript”,  “Voicemail
   Script”, “”, “ENU”, “USD”
   Question_PreLeave = ContinueOperation

End Function
```

Siebel eScript Code

```
function Question_PreLeave ()

{

   Script().Finish ();

   TheApplication().InvokeMethod ("RunSmartScript",

      "Voicemail Script", "", "ENU", "USD")

   return (ContinueOperation);

}
```

Calling Siebel Assignment Manager

Siebel SmartScript allows you to access Siebel Assignment Manager programmatically to determine the person that will handle a particular task. The following Siebel VB code is an example of how to call Siebel Assignment Manager from Siebel SmartScript.

Siebel VB Code

```
Dim SRBC as BusComp
```

```
Set SRBC = Question.GetSaveBusComp
SRBC.InvokeMethod “Assign”, “N”
```

NOTE: Before this function is called, the Save BusObj and the SaveBusComp business components must have been specified for the current question, and an instance of the business object must have been queried or created.

Importing and Exporting SmartScripts 8

SmartScript allows exporting and importing of scripts in a binary format between the development, testing, and production environments. In this guide, a file that can be imported or exported is referred to as a Siebel SmartScript file or .sss file.

Topics include:

Exporting Scripts

You can export scripts using a menu option available in the Scripts list on the SmartScript Administration screen. When you export a script, all the elements required for a script are copied to the destination file. When this file is imported into another Siebel database, the script should have all the elements required to execute successfully. Any conflicts with existing elements in the new database are resolved during the import process.

To export a script

1. Navigate to Site Map > SmartScript Administration
2. Select the script you want to export in the Scripts list, then click the Scripts menu and select Export Script.
3. In the Export dialog box, select the appropriate options regarding the data you are exporting.

   A File Download dialog box appears.
4. Click the hyperlink, and then select the location where you want the file saved.
5. In the Save As dialog box, enter the name of the file in which the script will be stored.

   You may use any file name, and it will have no impact on the SmartScript definition that will be imported from this file. It may be necessary to select open from the first window, and then save from the second window. You should always make sure that the file is saved with the extension .sss.
6. Click the Scripts menu and choose Save Record.

   The selected script will be saved to the specified file and location.

Importing Scripts

You can import scripts using a menu option available in the Scripts list on the SmartScript Administration screen. The import process checks many characteristics of script elements for error conditions. For example, it checks references to make sure that questions specified as branch destinations have been defined. When the import process has finished, a log file listing any errors can be viewed by clicking a link.

The import process transfers everything that does not cause an error or a conflict from the .sss file into the database. When you import a script, the script to be created or updated will be based on the script name for the exported script, not the file name for the script or the name of the selected script when the Import button is selected.

NOTE: When importing a SmartScript into a local SQL Anywhere Database in which pages for the importing SmartScript already exist, delete the preexisting pages to make sure that duplicate page branches are not created. This is necessary only when importing into an SQL Anywhere Database.

Once a SmartScript has been imported, verify it by using the Verification Wizard, and then perform normal testing.

To import a script

1. Navigate to Site Map > SmartScript Administration.
2. Click the Script menu and select Import Script.
3. In the Import Script form, click the select button, the fill in the required information, then click Add.

   If the “In Case of Error” drop-down appears, select how you want to handle conflict resolution. See “Resolving Conflicts Encountered During Import” on page 150 for details.

4 Click Import File in the Import Script list.

If successful, you will see a message indicating that the file was imported successfully, a log file with the details of the conflict resolutions, and a message list of the different elements that were processed as part of the import.

Resolving Conflicts Encountered During Import

If SmartScript encounters conflicts during the import process, SmartScript refers to the options you select in field titled "In case of error:" in the Import Script list. You may select from the following options: Update, Skip, Add.

These options determine how SmartScript handles the conflict or error. You can add the element, skip the import of that element, or update the existing element in the database.

If errors or conflicts were encountered, SmartScript creates a log file. This log file can be viewed when the import process has finished by clicking a link. Review the conflicts and errors reported there and correct each element as necessary, using the appropriate SmartScript Administration view.

If the contents of a picklist that appears in both the exporting and importing databases are different, a conflict will occur. The contents of the picklist should be updated to match in the databases prior to exporting the script.

9 Modifying the Dashboard

This chapter provides an overview of the Customer Dashboard, and the way in which you configure the SmartScript player to invoke the Dashboard and display for the user.

For more information about modifying and configuring the Customer Dashboard, see *Siebel Tools Reference*.

About the Customer Dashboard

The Customer Dashboard provides employees with persistent access to customer data, such as contact name, phone number, and interaction history. This data remains in the same location in the screen at the top, as long as the session is active and the dashboard is not closed. The Dashboard may also be active when the Search Center or Siebel-supported communications channels are active.

Example

The employee on an outbound telemarketing campaign logs into a predictive dialer using the Siebel Communication toolbar. Upon logging into the campaign, the predictive dialer begins contacting individuals and filters out answering machines, no answers, and busy signals.

While the employee is greeting the caller, SmartScript opens and the appropriate SmartScript for the campaign appears. By the time the employee has concluded greeting the caller, the initiated SmartScript page is read and the employee has the appropriate data to continue the interaction.

If at any point in time, the employee forgets key information about the customer or contact (for example, name, phone number, and so on), the employee can refer to the Dashboard.

If a call center employee regularly needs additional information about the contact, the Dashboard can be customized to provide access to different views, so that the employee can navigate to information related to the active customer.

If an employee regularly needs to search information, the Dashboard can be customized such that the results of the search populate the Dashboard. With the customization mentioned in the preceding paragraph, the employee can then navigate to information related to the search results.

If the caller gets accidentally disconnected, the employee can initiate an outbound call to reestablish the connection.

Overview of Customer Dashboard Configuration

The Customer Dashboard is preconfigured to capture and display information about the contact with whom communication is taking place, such as the contact name, phone number, address, and account. The values in this dashboard can be set through Communications Server events (such as CTI events), through the Search Center, through SmartScript, or directly through VB.

Typical tasks to change the preconfigured Dashboard include:

- Designing the customer dashboard layout with Siebel Tools. The customer dashboard displays an applet based on a special VBC.
- Linking business component fields to Customer Dashboard fields using Siebel Tools.
- Integrating the Communication Server with the Customer Dashboard. Communications Server Administrators use the Communications Administration screen. This involves work with communications commands and events.

For more information about working with Siebel Tools, see *Siebel Tools Reference*.

For more information on Siebel Communications Server, including information on using CTI to configure the Customer Dashboard, see *Siebel Communications Server Administration Guide*.

Overview of Upgrade to the Customer Dashboard

In Siebel 7, the Customer Dashboard replaces the SmartScript Dashboard from previous releases. The Customer Dashboard can be automatically updated from a SmartScript by following these steps:

- Make sure the SmartScript Player is configured to notify the dashboard. The default state of the SmartScript Player Applet is TRUE, allowing for this notification. See "To access SmartScript Player applet object" on page 155 for instructions.
- Make sure that updated parameters are configured in Siebel Tools to reflect the updated changes. See "To make sure updated parameters are configured in Siebel Tools" on page 156 for instructions.
- Update the Customer Dashboard by saving answers to dashboard variables. See "To update to the Customer Dashboard" on page 155 for instructions.

To update to the Customer Dashboard

1. Navigate to Site Map > SmartScript Administration > Questions.
2. Select a question, and then click the More Info view tab.
3. In the Save User Parameters field, enter the name of the parameter as it is listed in the SmartScript List in Siebel Tools.

   For example, “Fname”.
4. From the Show drop-down list, select Scripts, and then click the Translations view tab.
5. In the Dashboard Text field in the Translations list, enter the names of the variables from each question enclosed by brackets. For example: [Fname].

   This notifies the SmartScript to pass this parameter to the dashboard. Only parameters that have been configured in the Dashboard Business Service should be entered in this field. You must also make sure that the SmartScript Player is configured to notify the dashboard. The default state is TRUE. For more information see *Siebel Tools Reference*.
6. Click Save.

To access SmartScript Player applet object

1. Lock your project in Siebel Tools.
2. In the Object Explorer, choose Applet, then query for “SmartScript Player Applet (Tree Only).”
3. Click on the Applet User Property.

   The default state of this property is “Y”. Set the Value to “N” if you want to disable this feature.

To make sure updated parameters are configured in Siebel Tools

1. Lock your project in Siebel Tools.
2. In the Object Explorer, choose Business Service, and then query for "Persistent Customer Dashboard."
3. Click on the Business Server User Properties to see the current dashboard configuration.
4. Make sure that the parameters you entered in "To update to the Customer Dashboard" match identically the values in this list.

   This list is specifically used for updating the dashboard from a SmartScript. You may also update the dashboard from a SmartScript using Siebel VB or Siebel eScript. You must call the Persistent Customer Dashboard business service and pass the appropriate parameters.
5. To commit your changes, click anywhere outside the modified row, or move outside the row with the UP or DOWN arrow.
6. Compile your changes to the SRF file.

SmartScript Tags

A

Script Tags for SmartScript 7x

The Smart Script tag definitions are explained using example of the template file - CCSmartScriptPlayerApplet.swt.

NOTE: All tags are displayed in bold type.

```
<!- Start of CCSmartScriptPlayerApplet.swt ->


<!-- Start of Script title -->

<table width="98%" border=0  cellspacing="0" cellpadding="0"
align="center">

<swe:form name="SmartScriptForm">

<!-- Start of Script Section-->

<!-The block enclosed by the <swe:control id="SmartScriptLabel"> is
used to display any script information on the HTML page

swe:control - generic SWE tag. Provides a placeholder in an Applet
Web Template for a control object or a list column object. The
control could either be mapped in Tools or defined at runtime.

     Id -SmartScriptLabel: The id uniquely identifies that control
     as a placeholder for the translated label of the smartScript

<tr>

     <td width="66%">

     <div class=CmdTxt><swe:control id="SmartScriptLabel"
     ><div><p></td>

     <td width="33%"> </td>

</tr>

</table>

<!-- End of script section -->
```

```
<!-- Start of Pages section -->

<!-The tag <swe:control id=SSPageLabel> is used to display the
current active page ->

swe:control - a tag that's used as a placeholder for a control
(either mapped in Tools or defined at runtime)

     Id- SSPageLabel: the id uniquely identifies the control as a
     placeholder for the translated label of the page

<table class="AppletStyle1" width="98%"
cellspacing="0"cellpadding="0" border="0" align="center">

<tr>

<td>

     <table width="100%" cellspacing="0" cellpadding="0" border="0"
     align="center">

     <tr class="AppletBlank">

     <td class="AppletTitle" valign="top" width="8"><img
     src="images/aplt_opn_d.gif" width="8" height="10" border="0"
     alt="" /></td>

     <td colspan="5">

        <table cellspacing="0" cellpadding="0" border="0">

        <tr>

        <td class="AppletTitle"><nobr><swe:control
        id="SSPageLabel"/></nobr></td>

        <swe:control id="6">

        <td class="AppletButtons" valign="middle" nowrap>

        <swe:this property="FormattedHtml" hintText="Save Answers"
        hintMapType="Control"/>

        </swe:control></td>

        <td class="AppletTitle" width="22"><img src="images/
        aplt_curve_d.gif" height="18" width="22" border="0"
        alt=""></img></td>
```

```
            <td width="100%" class="AppletBlank" align="right">

            <span class=required>*</span>

            <swe:control id=1500 property="DisplayName"
            hintText="Required Label" hintMapType="Control"/></td>

            </tr>

            </table>

         </td>

         </tr>

         </table>

</td>

</tr>

<tr>

<td class="AppletStyle1">

         <table width="100%" cellpadding="2" cellspacing="0" border="0"
         align="center">

         <tr>

         <td class="AppletBorder">

            <table valign="top" width="100%" cellpadding="2"
            cellspacing="0" border="0" class="AppletBack">

            <tr>

            <td colspan="3"><img src="images/spacer.gif" width="1"
            height="3"></td></tr>



            <swe:error>

            <tr>

            <td width=20%> </td>

            <td class="error" width=75%>
```

```
<swe:this property="FormattedHtml"/></td>

<td width=5%> </td>

</tr>

<tr>

<td colspan="3">

<img src="images/spacer.gif" width="1" height="3"></td></tr>

</swe:error>

</table>


<!--Start of questions in the current section on the Current Page -->

<!-The block enclosed by the <swe:for-each id="SSQuestionList"
Count="Dynamic" StartValue=1000 IteratorName="SSQuestionIndex">
encloses all questions displayed within the current page -->
```

swe:for-each- generic SWE tag used as an iterator.

Id = SSQuestionList: Identifier that uniquely identifies this iterator to be pertaining to that of SmartScript questions

Count = Dynamic: Denotes that the number of times the tag should iterate its contents. This is dynamically determined at runtime depending on the number of questions that need to be displayed.

StartValue = 1000: Indicates the value at which the iteration will start. The tag will start the iteration by assigning this value to an internal iterator and will increment it by one for each iteration.

Iterator Name = SSQuestionIndex: A name for the iterator. This name can be used to get the value of the iterator during the iteration by using the syntax swe:iteratorName.

<!-The block enclosed by the <swe:control id=SSQuestion> is used as a template for all questions other than Information type questions-->

swe:control- generic SWE tag. Provides a placeholder in an Applet Web Template for a control object or a list column object.

Id = SSQuestion Identifies the control as a question.

swe:this - Refers to the object inside which it is placed.

Property = RequiredIndicator: If the displayed question is mandatory then using this tag displays the default required indicator used within the system.

Property = DisplayName: Shows the title of the question

Property = FormattedHtml: Shows the control for the question

<!-The block enclosed by the < swe:control id=SSInfoQuestion > is used as a template for all information type questions-->

swe:control- generic SWE tag. Provides a placeholder in an Applet Web Template for a control object or a list column object.

Id = SSInfoQuestion Identifies the control as a question.

swe:this - Same as above

<!-Start of questions block->

<swe:for-each id="SSQuestionList" Count="Dynamic" StartValue=1000 IteratorName="SSQuestionIndex">

<swe:control id="SSInfoQuestion">

<table valign="top" width="100%" cellpadding="2" cellspacing="0" border="0" class="AppletBack">

<tr>

<td width=20%> </td>

```
        <td width=75%><swe:this property="DisplayName"/></td>

        <td width=5%> </td>

        </tr>

        <tr>

        <td colspan="3"><img src="images/spacer.gif" width="1"
        height="3"></td></tr>

        </table>

</swe:control>


<swe:control id="SSQuestion">

        <table valign="top" width="100%" cellpadding="2"
        cellspacing="0" border="0" class="AppletBack">

        <tr>

        <td width=20%> </td>

        <td width=75%>

        <div class="scLabel">

        <swe:this property="RequiredIndicator"/>

        <swe:this property="DisplayName"/>

        </div>

        <span class="scField">

        <swe:this property="FormattedHtml"/></span>

        </td>

        <td width="5%"> </td>

        </tr>

        </table>
```

```
</swe:control>


</swe:for-each>

<!-- End of questions block -->


<!-Page Divider-->

<table valign="top" width="100%" cellpadding="2" cellspacing="0"
border="0" class="AppletBack">

<tr>

<td width=20%> </td>

<td width=75%> </td>

<td width=5%> </td>

</tr>

<!-- End of page divider -->


<!-Start of Buttons -->

<tr>

<td> </td>

<td>

      <table cellpadding="0" cellspacing="0" border="0">

      <tr>

      <!-- NextSection -->

      <swe:control id="4">

      <td valign="middle" nowrap>

      <swe:this property="FormattedHtml" hintText="Next Section"
      hintMapType="Control"/>
```

```
</td>

</swe:control>


<!-- PreviousSection -->

<swe:control id="3">

<td valign="middle" nowrap>

<swe:this property="FormattedHtml" hintText="Previous Section"
hintMapType="Control"/>

</td>

</swe:control>


<!-- Finish: 1 -->

<swe:control id="1">

<td valign="middle" nowrap> <img src="images/
btn_div.gif"> </td>

<td valign="middle" nowrap>

<swe:this property="FormattedHtml" hintText="Finish Script"
hintMapType="Control"/>

</td>

</swe:control>


<!-- Save: 5 -->

<swe:control id="5">

<td valign="middle" nowrap>

<swe:this property="FormattedHtml" hintText="Save Script"
hintMapType="Control"/>

</td>
```

```
        </swe:control>


        <!-- Cancel: 2 -->

        <swe:control id="2">

        <td valign="middle" nowrap>

        <swe:this property="FormattedHtml" hintText="Cancel Script"
        hintMapType="Control"/>

        </td>

        </swe:control>

        </tr>

        </table>

</td>

<td> </td>

</tr>

<!-- End Buttons-->


<!-- Page divider -->

<tr>

<td> </td>

<td> </td>

<td> </td>

</tr>

<!-- End of page divider -->

</table>

</td>
```

```
</tr>
</table>
</td>
</tr>
</swe:form>
</table>
<!-- End of CCSmartScriptPlayerApplet.swt-->
</swe:form>

<!-- End of CCSmartScriptPlayerApplet.swt-->
```

Index

C

D

E

F

G

H

I

J

M

N

O

P

Q

R

S

T

U

V

W