Siebel SmartScript Administration Guide

Siebel Innovation Pack 2017, Rev.A
November 2017

ORACLE

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Contents

Chapter 4: Working with Questions, Answers, and Translations

Chapter 5: Upgrading SmartScripts

Chapter 6: Customizing a SmartScript User Interface

Chapter 7: Verifying, Testing, and Invoking SmartScripts

Chapter 8: Extending Scripts with Siebel VB and Siebel eScript

1 What's New in This Release

What's New in Siebel SmartScript Administration Guide, Siebel Innovation Pack 2017, Rev. A

This guide has been updated to correct or remove obsolete product and component terms.

What's New in Siebel SmartScript Administration Guide, Siebel Innovation Pack 2017

Table 1 lists the changes in this revision of the documentation to support this release of the software.

NOTE: Siebel Innovation Pack 2017 is a continuation of the Siebel 8.1/8.2 release.

Table 1. What's New in Siebel CTI Administration Guide, Siebel Innovation Pack 2017.

Topic	Description
Invoking SmartScripts Using a Hyperlink on page 65	Modified. Siebel Tools changed to *local*
To make sure the updated parameters are configured in Siebel Tools on page 125	Modified. SRF changed to Siebel runtime repository

What's New in Siebel SmartScript Administration Guide, Siebel Innovation Pack 2016

No new features have been added to this guide for this release. This guide has been updated to reflect only product name changes.

What's New in Siebel SmartScript Administration Guide, Siebel Innovation Pack 2015, Rev. A

A note has been added to this guide regarding a change in the software used with the local database for Siebel Remote.

2 Overview of Siebel SmartScript

This chapter provides an overview of Siebel SmartScripts. It includes the following topics:

About Siebel SmartScript

Siebel SmartScript allows business analysts, call center managers, and Siebel developers to create scripts to define the application workflow for interactive customer communications. The script determines the flow of the interaction, not 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 the call center operations.

You can use Siebel SmartScript with customer applications, allowing the same troubleshooting scripts to be shared between the customers using the Siebel Self ServiceSelf Service site, and the 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.

Oracle's Siebel business customer communications include marketing applications where customers interact with Web surveys, or email questionnaires, or customers using interactive troubleshooting guides on Service Web sites, such as Siebel Self Service.

Benefits of Using Siebel SmartScript

Siebel SmartScript offers the following benefits:

- **Software-controlled workflow.** Siebel 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. Siebel SmartScript selects the appropriate branch of questions as needed. Branching activity is based on the answers and responses that the customer or prospect selects.
- **Reduced training time.** Siebel 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.** Siebel SmartScript allows business analysts and call center managers, rather than systems analysts or programmers, to design and implement the workflow. This capability 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 capability allows a user session using the script to be personalized and effective.
- **Dynamic updating.** Using branching logic, Siebel SmartScript displays only those questions in a script that are pertinent to a given transaction.
- **Search Center integration.** Smartscripts can be searched and displayed in the Search Center results field. The user can then click the hyperlink and launch the SmartScript.
- **Dashboard.** Information gained from a SmartScript can be dynamically updated on the customer dashboard so it can be viewed during the customer interaction.
- **Efficient modification and reuse of scripts.** Scripts are built from modular elements: questions, pages of questions, answers, and branching instructions. You can use an element, such as a page of questions—or even an individual question— in multiple scripts or in multiple language translations for a single script.

Prerequisites for Using SmartScripts

To use Siebel SmartScript effectively, you must fulfill the following prerequisites:

- You must be familiar with Siebel user interface standards.
- If you want to use advanced scripting to extend the standard capabilities of Siebel 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 scripting language like JavaScript) can be used with all SmartScript elements.
- If you want to deploy SmartScripts within a multilingual call center, you need translations for all script elements in each language in which the script runs.

- If you want to deploy graphically customized SmartScripts, such as SmartScripts that are graphically integrated with existing Web sites, you must have HTML editing and Web site design experience.

About Scripting Terminology

You must be familiar with the terms listed in Table 2 to understand scripting.

Table 2. 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, and the branches needed to move between the pages.
page	The logical grouping of questions within a page that display together for the user.
question	A 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	A 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, drop-down lists, and multi-value groups.
translation	A text string used to display script elements in languages other than the original, so that scripts can be used in multilingual call centers. The 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	The transfers of control inside a SmartScript that define the display and processing sequence of pages or questions.
page section	The logical grouping of questions within a page that 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 while 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 that 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.

About Siebel SmartScript Screens

Siebel 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.

This section includes the following topics:

About 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 that allows the user to navigate through the script, and the SmartScript Player where the 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 Siebel SmartScript, a list appears from which the agent clicks 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 are displayed.

After 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 if they want to restart it later.

About the SmartScript Explorer

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

The SmartScript Explorer allows viewing and navigating through a script in a more dynamic, flexible way. It includes the standard Windows plus sign (+) and minus sign (–) icons that signify whether additional information can be displayed. Different icons have different meanings:

- Check mark icons (in Green) signify questions that you have answered satisfactorily.
- Question mark icons (in yellow) signify optional questions or prompts.
- Question mark icons (in Red) signify mandatory questions.

The SmartScript Player on the screen (right side) automatically generates the questions and information text 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.

About the Customer Dashboard

Displayed at the start 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. For more information on updating the Customer Dashboard from previous releases of Siebel applications, see Chapter 10, "Modifying the Customer Dashboard". For more information about the Customer Dashboard, see *Configuring Siebel Business Applications*.

About 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 abandoned. Users can resume scripts from the My Saved Sessions view.

About 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 a 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 Consumer Marketing application. SmartScript questions and answers are displayed side-by-side.

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

About the Administration - SmartScript Screens

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

Siebel SmartScript uses objects that are generically called *elements* to create the business process flow. The Administration - SmartScript 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 3, "Fundamentals of Creating SmartScripts."

About Deploying SmartScripts to Remote Users

This topic describes remote access and replication management of SmartScripts.

A remote user on a Siebel Mobile Web Client can use a SmartScript. If a SmartScript administrator changes a SmartScript, the Siebel Remote Manager synchronizes these changes for the remote user.

Note the following points if you intend to deploy SmartScripts to remote users:

- You can specify a replication level for a SmartScript question.

  To specify the replication level for a question, select a value in the Replication Level drop-down list of the More Info form in the Questions view of the Administration - SmartScript screen. The available values are as follows:

  - **All.** By default, questions have a replication level of *All* that allows the replication of questions to all regional databases and the local databases of remote users.
  - **Regional.** Specifies that you can replicate questions to regional databases only.
  - **None.** Specifies that you cannot replicate questions from the server database.
- By default, SmartScript elements are defined as dock objects with a visibility level of Limited for remote access management. For dock objects with Limited visibility, Siebel Remote sends the object to a Mobile Web Client only if the instance is visible to the remote user.

For more information about remote access and replication management, see the *Siebel Remote and Replication Manager Administration Guide*.

3 Fundamentals of Creating SmartScripts

This chapter describes the fundamentals of creating SmartScripts. It includes the following topics:

About SmartScript Elements

Questions are the main element or object inside a SmartScript and include such attributes as answers and data indicating how a question is displayed. Questions are joined together with branches. 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 the 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 8, "Extending Scripts with Siebel VB and Siebel eScript."

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

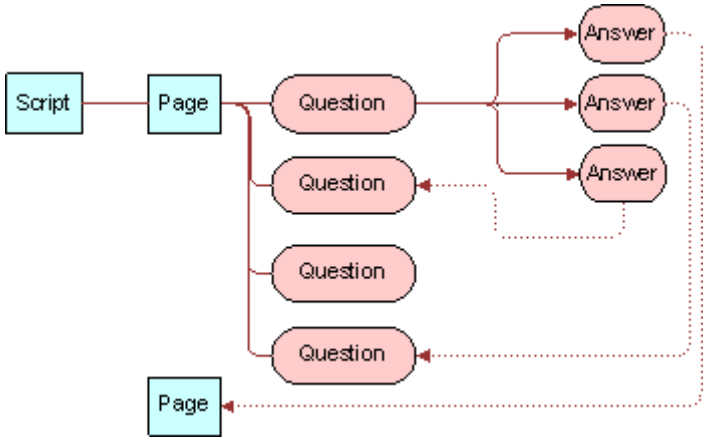

Figure 1. Scripts in the Hierarchy of Script Elements

Table 3 describes each script element in more detail.

Table 3. SmartScript Elements

Element	Description
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 must contain related questions, and questions must 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 elicit information from the caller or Web user. Questions also provide information about policies, options, and product descriptions. ■ 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. Answers are not stored for information questions.
Answers	Answers are script elements that represent answers to questions. Answers appear as data entry fields or as one 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.

Table 3. SmartScript Elements

Element	Description
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 might ask whether a customer is making a deposit to, rather than a withdrawal from, an account. The possible answers require different subsequent questions. Therefore, you must create a separate branch for each possible answer. At other times, a SmartScript user might need to interrupt the expected flow of a script. For example, if a loan agent makes a call to offer a credit card but discovers that the target works for a competing bank, it makes no sense to continue with the standard script; end the call as quickly, and as smoothly as possible by branching to the end of the script. **CAUTION:** There are no built-in safeguards to prevent you from creating branches that result in closed loops of questions. Therefore, you must test all your branches to be sure that they have an exit.

About Data Storage in Siebel SmartScript

You can store answers gathered using SmartScripts in a number of ways. You also have the option of not storing questions at all. Storage methods are listed as follows:

- **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 *Configuring Siebel Business Applications*.

If a script's Save Session parameter is set to Finished, Always, or Finished/Timeout Save, and a question's Save Answer Table check box is selected, the script session information, including the questions and answers, can be viewed in the Script Sessions view of the Administration - SmartScript screen.

Store answers in the business component fields when the SmartScript is collecting data that is represented in the system's business objects. For example, the caller information collected during a script-controlled transaction must 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 to follow. For example, a consumer information center might supply both information about rebate offers and answer questions about product safety. The question of which of these topics is the reason for a call might not be relevant to the particular call record, but the answer probably determines which pages of the script must be displayed.

NOTE: The Save Session parameter and Save Answer Table check box cannot be used independently of each other. They work together to save answers to the Sessions Tables. They are also independent of Save Bus Comp, Save Field, and Save Bus Object. A setting cannot affect when the answers are saved to the Save Bus Comp fields.

Viewing SmartScripts

This topic describes how to view SmartScripts in an employee application, or as an administrator.

To view saved SmartScripts in an employee application

- From the employee application home page, navigate to SmartScripts screen > My Saved Sessions view.

  The list of saved SmartScript sessions for the employee appears.

To view saved SmartScripts as an administrator

- Navigate to Administration SmartScript screen > Script Sessions view.

  You can restart a script from this view that has been saved using the Finish Later button. If the contact ID is set during the script, it is saved in the script sessions table. As a result, a script session can be viewed because of its association with a contact in the Contacts screen.

Guidelines for Creating SmartScripts

Create all your questions and answer elements first, then create the structure of your script. Perform these steps in the following order:

1. Understand your business scenario, and map out your design.
2. Create questions and translations. For more information, see “About Creating Questions” on page 31.
3. Create answers. For more information, see “Creating Answers” on page 40.
4. Create pages.
5. Create the script.

6 Add questions to pages.

7 Add pages to the script.

For more information, see the following topics:

- "Script Design Tips" on page 21
- "Creating Pages" on page 21
- "Translating Pages" on page 22
- "Creating Scripts" on page 22
- "Translating Scripts" on page 24

Script Design Tips

While detailed script design is outside the scope of this guide, use the following general guidelines when you create scripts:

- Use a conversational writing-style that seems to be part of a natural dialogue between the agent and an interested customer or prospect.
- Use a flexible script that allows the agent to respond to unforeseen questions, comments, and objections.
- Use a general script that addresses the needs of all potential customers or prospects, without forcing them into preexisting categories that might not match their needs.

Creating Pages

You create pages in the Pages view of the Administration - SmartScript screen.

To create a page

1 Navigate to Administration - SmartScript screen > Pages view.

2 Create a new record.

3 Enter the name of the page in the Names field.

4 In the First Question field, select the question that you want to appear first on the page.

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 runs.

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 Administration - SmartScript screen > Pages view.
2. In the Pages list, select the page you want to translate.
3. Click the Translations view tab and create a new record.
4. In the Language field, select a language.
5. In the Label field, enter the translation of the page's title.

Creating Scripts

A script is the root element of a SmartScript. Scripts contain pages that contain questions. Questions contain question translations and answers. Branches between elements define the structure of the script within which the elements appear. You create scripts in the Administration - SmartScript view.

To create a script

1. Navigate to Administration - SmartScript screen > Scripts view.
2. Create a new record, and complete the necessary fields.

   Some fields are described in the following table.

Field	Description
Script	Enter a name for the script.
Type	Select the script type to limit the SmartScripts that display in particular applications. For example, the scripts that Self Service uses must be of type Instructions or Troubleshooting.
First Page	Select the first page to appear in the script. Alternatively, you can create a new first page from the Pick Page dialog box that appears.

Field	Description
Save Session	Select one of the following values: ■ 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 Save 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.
Jumping Allowed	Allows users to navigate in any order through the script using the SmartScript explorer or the Previous button.
Business Component	SmartScripts to be invoked from the Script button on the Account, Contact, or Opportunity screens must have this field set to one of those business components. When the Script button is selected, the SmartScript that has the BusComp value equal to the business component of the current applet invokes automatically.
Description	A description of the SmartScript. You can use this description to build indices of SmartScripts.
Estimated duration	Reference field for how long a script is expected to take to complete.
Duration In	Units for the estimated duration.
Organization	Specify the organization whose members have access to the script.

Field	Description
On Cancel Go to View	View to go to when the Cancel button is selected.
On Finish Go to View	View to go to when the Finish button is selected. If you define views in the GotoView method for the Script_Finish event, make sure that the event logic accommodates the view that you designate in this field. For example, the Script_Finish event might facilitate navigation to different views based on different conditions. If none of the conditions exist, then the event can facilitate navigation to the view that you designate in this field. If a conflict exists between the view that you designate in this field and the view that the Script_Finish event designates, the view that you designate in this field is in effect.

3 Create translations for the script title as described in "Translating Scripts" on page 24.

4 Attach other pages to the script as described in "Adding Pages and Branches to Scripts" on page 27.

Translating Scripts

Script translations are translations to different languages of the individual Script titles displayed the SmartScript screen.

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

To create a translation for a script title

1 Navigate to Administration - SmartScript screen > Scripts view.

2 Select the script whose title you want to translate.

3 Click the Translations view tab, create a new record, and complete the necessary fields as described in the following table.

Field	Description
Language	Select a language.
Label	Enter the translation of the script title.
Dashboard Text	Enter the text and parameters to appear in the dashboard. For more information, see Chapter 10, "Modifying the Customer Dashboard." **NOTE:** The Customer Dashboard is available only in employee applications.

Field	Description
Released	The value in this field is generated by the application when you release a script. If the check box is selected, it indicates that the translation and its associated script have been released. For more information, see "About Releasing SmartScripts" on page 28.
Date	The value in this field is generated by the application when you release a script. It indicates the date when the translation and its associated script were released. For more information, see "About Releasing SmartScripts" on page 28.

About the Script Designer and Page Designer

Use the Script Designer and Page Designer to create scripts using a graphical user interface (GUI).

The SmartScript Script Designer and the SmartScript Page Designer screens are similar. The Designer view workspace appears towards the end of both screens. The palette provides page or question icons (depending on which designer view) and branch icons in both views that allow you to join the different elements. You drag script elements onto the workspace to create your scripts. A list of pages appears on the Page Designer screen while the SmartScript Script Designer screen displays a form with details of the current script.

After 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 connect to display a large box. In the workspace, you can double-click 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 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 persists 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 extent of the workspace.

Adding Questions and Branches to Pages

You can add questions and branches to pages to design the flow for the page. You can add each question or branch element using the drag-and-drop (copy and paste) GUI of the Page Designer.

To add questions and branches to a page

1. Navigate to Administration - SmartScript screen > Pages view.
2. 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 is already selected in the Pages list applet.

4 Copy and paste the question icon from the Page Designer palette to the workspace.

The Pick Question dialog box appears.

5 Select a question from the list, and click OK.

6 Repeat Step 4 and Step 5 to add another question.

7 Drag a branch icon from the palette to connect these two questions with a branch.

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 to select an answer for the branch to use 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.

8 Drag the arrow end of the branch to align it with a connection point on the question to which you want to branch.

To view the branches within a page

1 Navigate to Administration - SmartScript screen > Pages view.

2 Click the Branches view tab, and select All Branches from the Visibility filter.

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 as described in "Adding Questions and Branches to Pages" on page 26.

To add pages and branches to a script

1 Navigate to Administration - SmartScript screen > Scripts view.

2 Select the script into which you want to add or modify pages.

3 Click the Designer view tab.

The workspace for the selected script appears.

4 Drag the page icon from the Script Designer palette to the workspace.

The Page Pick dialog box appears.

5 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 Pick Question 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 Pick Question dialog box appears.

4. Select a question from the list and click OK.
5. To continue adding the required branches and pages to the script, repeat Step 1 to Step 4

To view the branches between pages

1. Navigate to Administration - SmartScript screen > Scripts view.
2. Click the Branches view tab, and select All Branches from the Visibility filter.
3. The Branches list appears and displays all branches between the pages in the script.

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

About Releasing SmartScripts

After 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. It is also recommended that you release scripts if you intend to save a script during execution and resume its execution at a later time. If you do release the script, it will not be possible to resume it from a saved session.

Release scripts only when you are ready for production. Otherwise, you must release scripts again after each change made so that the change is visible when testing.

Releasing a script saves a precompiled version of the SmartScript definition and all the associated code (VB or eScript). Releasing a script 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 creates a pointer to the release file in its server location.

When a user invokes a released script, the SmartScript engine looks 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 copies a release copy of the script from the file system to the server and then executes 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 generated. If a script is not released, Siebel SmartScript must compile and execute the script from the server, which causes the script to load more slowly.

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

About the Release Process

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

1. The released file is created and put on the Siebel File System under the Siebel File System root directory; no subdirectory exists. This released file is saved as a SAF file, which is compressed in the same manner as all the other Siebel File System files are compressed.
2. If a client starts the SmartScript, the compiled version is downloaded from the Siebel File System to the following directory on the server machine:

   `[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 make sure you delete the released version, and release the revised script again. Otherwise, Siebel SmartScript continues to use the previously released version of the script, which does not include the most recent changes.

Releasing Scripts

This topic describes how to release a script.

To release a script

1. Navigate to Administration - SmartScript screen > Scripts view.
2. Select the script you want to release.
3. Click the Translations view tab and select the translation you want to release.
4. From the Scripts menu click 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.

Deleting the Released Version of a Script

Deleting the released version of a script erases the pointer to the script file. Each subsequent execution of the script is compiled from the database every time you run the script. This mode is the same as the development mode.

To delete the released version of a script

1. Navigate to Administration - SmartScript screen > Scripts view.
2. Select the script that you want to delete.

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

4 From the Scripts menu, select Unrelease.

The released version of the script is deleted.

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

4 Working with Questions, Answers, and Translations

This chapter explains how to create questions and answers in the Questions view of the Administration - SmartScript screen. It includes the following topics:

About Creating Questions

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

Questions can serve various functions in your SmartScripts such as:

- **Eliciting information.** A call center agent can ask questions to elicit sales or service information from a customer. Over the Web, questions can be presented to a customer as a survey or a series of questions to isolate a problem. The answers given can then be stored for later use.
- **Providing information.** A question can provide text for an agent to read to a customer, or a customer can read it 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 SmartScript is to 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 More Info form of the Questions view in the Administration - SmartScript screen 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 Comp, and Save Field all serve to define the location for answer data given in response to a question. The fields labeled Width, Height, Min Length and Max Length all refer to the user interface space provided for answers.

For more information about creating questions, see the following topics:

Creating Questions

This topic describes how to create a question.

To create a question

1. Navigate to Administration - SmartScript screen > Questions view.
2. Create a new record, and enter the question's attributes in the More Info form.

   The following table describes the attributes a question can have.

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 want to use: ■ Check box—Single-select (Boolean) and multiselect check boxes ■ Default The default answer control used depends on the associated answer type. ■ Dropdown—Displays the 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.

Question Setting	Description
Min Length and Max Length	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 the following: ■ The minimum restriction you specify for Min Length triggers only when you try to edit the question value irrespective of whether you are creating a new record or updating an existing one. ■ 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 is one-half of the number entered in these fields.
Auto Sub Parm	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 might be a user parameter set 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 is converted to the current value for that parameter. **NOTE:** The [BC.Field Name] works only if the SmartScript is positioned on the relevant Business Component. If the user has positioned on a business component by scripting behind the SmartScript, no data is 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 SmartScript. ■ Always required—Is the strictest answer setting, because the user cannot finish the SmartScript unless the 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 must be selected from a pick applet, then the default answer is not displayed.
Width	The width, in pixels, of the text box provided for answers.
Height	The height, in pixels, of the text box provided for answers.

Question Setting	Description
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 business component 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 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 *Using Siebel Tools*.
Save Business Object	The business object in which the answer to the question is stored.
Save Bus Comp	The business component in which the answer to the question is stored. Note that to save a question's answer to a business component field, the business object, the business component, and the field name must be specified for the question. For more information, see "Displaying Answers Within a SmartScript" on page 37.
Save Field	The field in the business object or business component table that is to contain the answer data, or that is used to identify the specific record that contains the answer data. Note that special steps might be required to set up a save field. For more information, see "Displaying Answers Within a SmartScript" on page 37.

Question Setting	Description
Save User Parameters	Specifies a field in a business component as a user parameter. The record that is set as active in that business component 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 business component 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 contains the currency setting.
Pick Applet	Indicates the pick applet that the end user uses to select and save the 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 37.

Question Setting	Description
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 37.
Detail Applet	Indicates the Detail applet that is 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 37.
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

This topic describes how to define question text and translation text. In Siebel SmartScript, the text of each question is treated as a translation regardless of the language used. Even if you create only 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: After you have decided to translate the text into a particular language, you must translate all questions in the SmartScript into that language. If you do not translate every question within the SmartScript, your SmartScript might not run properly.

For more information on creating and translating questions, see Chapter 3, "Fundamentals of Creating SmartScripts."

To create a question's translation

1. Navigate to Administration - SmartScript screen > Questions view.
2. In the Questions list, select the question that you want to define as question text.
3. Navigate to the Translations view tab, and create a new record.
4. In the Language field, select the language in which you create the question text, and click OK.

5 In the Question field, type the text of the question in the selected language.

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

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

Displaying Answers Within a SmartScript

You can define each question in a SmartScript to contain answer controls or be informational in nature. Informational questions do not require answer controls; they 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 that a user might select from, you have the option of defining these answer options within the SmartScript definition as Question Answers, or reusing an existing Pick applet or MVG applet already defined in Siebel Tools.

NOTE: SmartScripts do not support the use of association applets.

In the simplest case, users select answers to a question from a drop-down list of answers that you define specifically for that question, using the Answer subview of the Questions view in the Administration - SmartScript screen. These drop-down lists of answers are defined with respect to a unique question and are an extension of the question definition.

To use a drop-down list, select the Pick Only check box in the Questions view.

NOTE: By default, the Pick Only column does not appear. You must display it.

To use single or multiselect pick applets, or detail applets, do not select 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 of selecting from a fixed list of answer options, but instead types 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 SmartScript definition. You define a unique domain of answers for the question. Fields that are based on LOVs 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, because the user can choose only a single selection 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 check boxes. If you select Boolean for your answer type, only one check box appears.

Pick Applets

Any question can employ a Pick applet for users to select answers to that question. For a question to use a Pick applet or MVG applet, you must set the question to save the selected record to the appropriate field in the business component. To do this, you enter values for the following parameters:

- Save Business Object
- Save Bus Comp
- Save Field

You cannot define branching for answers selected from Pick applets. You must configure a Pick applet or MVG applet 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 Bus Comp. Selecting an appropriate applet requires some familiarity with Siebel Tools and the business components that SmartScript must be linked to. For more information on working with applets, see *Using Siebel Tools*.

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 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 Popup applet (SmartScript). If you use the Currency Popup applet, the agent sees an icon next to the question's input box at run time. 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. For a File Popup applet (SmartScript), the user can add a file to the record.

NOTE: Branching from a question can only be defined for answers that are defined as part of the SmartScript definition and not those selected from pick applets, or based on the list of values. If a user selects multiple responses to a question that uses multiselect check boxes, the default branch is used.

Answer Types and Answer Control Choices

Table 5 describes Answer Types and the Answer Control choices you have for each type.

Table 5. 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 is selected and answers exist) Text Box (if Pick Only is not selected) Pick/MVG/Detail Applet (if they are 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

Table 5. 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
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

You define answers to questions in the Answers subview of the Questions view in the Administration - SmartScript screen. You do not need to define answers if one or both of the following apply:

- Question presented is information for the end user
- Answer options come from a business component

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 required languages, as needed for each answer. Answer translations are optional. However, if you provide one translated answer within a SmartScript, you must translate every subsequent answer.

If an answer consists of numeric values (for example, where Answer Type is integer, number, or currency) do not translate the values, as long as the unit of measure is understandable to users who speak the target language. For information on creating and translating answers see Chapter 3, "Fundamentals of Creating SmartScripts."

To create an answer

1. Navigate to Administration - SmartScript screen > Questions view.
2. Select the question for which you want to define answers.

   If Pick Only is selected for the question, it can have more than one answer. You must specify all valid answers.
3. Navigate to the Answers subview, and create a new record.

4 Complete the fields as described in the following table.

Answer Form Field	Entry
Number	Provide a number to 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 includes 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 appears with this answer.

The new answer definition appears in the Answers subview.

5 Repeat Step 3 and Step 4 to add another answer.

"Translating Answers" on page 41 describes how to add translate answers.

Translating Answers

When the Answer Type for a question is String, and the Pick Only check box is selected, you might have to translate the answers so that they are displayed in the same language as the corresponding question translation.

NOTE: You do not require answer translations if you deploy the SmartScript in one language. If a SmartScript does not use translations, the language-independent answer value will be used for all translations. However, if you translate answers within your SmartScript, you must translate every question with answers into all the languages into which the SmartScript is translated.

To create a translation for an answer

1 Navigate to Administration - SmartScript screen > Questions view > Answers subview.

2 Select the answer you want to translate.

3 Navigate to the Answer Translations list, and create a new record.

4 Complete the necessary fields.

5 Repeat Step 1 to Step 4 for each additional language in which the answer must display.

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 can save header information about each SmartScript session and also specific questions and answers for the session. To do so, you must select the Save Session check box in the Scripts view of the Administration - SmartScript screen, and select the Save Answer Table check box for any question that you want to log as part of the Script Sessions information.

NOTE: You select the Save Answer Table check box in the Scripts view of the Administration - SmartScript screen.

This is useful when you do not want 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 SmartScript execution.

NOTE: You cannot use the Save Session and Save Answer Table independently of each other. They are designed to work together to save answers to the Sessions Tables. They are also independent of Save Business Object, Save Bus Comp, and Save Field.

The Call Script Runs and Call Script Run Answers business components are based on the Script Sessions table and the Answers table, respectively.

You can use the business components to store answers provided by the user during a SmartScript session. They allow you control whether a session is created and which questions have their answers saved in the answers table.

The Call Script Runs business component saves the following values:

- SmartScript name
- Date and time the SmartScript started
- Employee name if the SmartScript executes from an employee application
- Contact name if the SmartScript is run from a customer application

In addition, the duration of the SmartScript execution is automatically saved as well as the language in which the SmartScript 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 might get a single session record and multiple answer records associated with that session's record for every Siebel SmartScript run.

Saving to a Business Component

You can use the answers provided during a SmartScript session to update the existing records or create new records in any business component. You can specify the location to which Siebel SmartScript saves the answer data by completing the following fields in the More Info form of the Questions view in the Administration - SmartScript screen:

- Save Business Object
- Save Bus Comp
- Save Field

Fields that are mapped from drop-down lists must use pick applets. You can set the SmartScripts to either update an existing record or create a new one. The key logic occurs when a question is found that has a value specified in the Save Bus Comp and Save Field fields. If an active record is already set in that business component, then Siebel SmartScript determines that this is the record that you want to update. If an active record is not yet set in that business component, then a new record is created and the answers are 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 is saved to that field for the active record. If a pick applet is used, then it behaves 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 are not 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, do 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 Business Address business component of the Accounts business object. If the Business Address business component is not part of the Accounts business object, first add it, using Siebel Tools, to make it available. When setting up the question, enter Accounts in the Save Business Object field and Business Address (or any of the Address Multi Value Fields) in the Save Bus Comp 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. Navigate to the Questions view of the Administration - SmartScript screen, and create a new record.
4. In the Save Business Object field, pick the parent business object from the picklist.

5 In the Save Bus Comp field, pick the business component with which the multi-value field is associated.

6 In the Save Field field, select the multi-value field that you want to save the data from the picklist.

Question Events Logic Sequence

Figure 2 illustrates the logical order and structure of how Siebel SmartScript uses VB code and search specs, creates new records, and does text substitution. Table 6 provides a description of each label in Figure 2.

For more information on using Siebel VB or eScript to extend the SmartScripts, see "Extending Scripts with Siebel VB and Siebel eScript" on page 69.

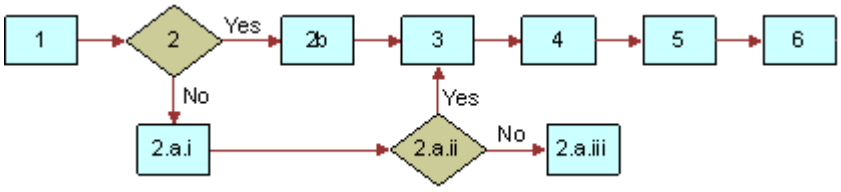

Figure 2. Logical Sequence of Question Events

NOTE: Each business component can have only one active record for a single user session, but there may be many active records in multiple business components for a single user session.

Table 6. Description of Labels in Figure 2

Label	Description
1	Question_Enter
2	Checks if a search spec exists for the question? If yes, Siebel SmartScript proceeds to step 2.a.i. If not, Siebel SmartScript proceeds to step 2.b.
2.a.i	Checks if the Save Bus Comp field contains a value.
2.a.ii	Checks if the business component specified in the Save Bus Comp field contains an active value?
2.a.iii	Creates a new record in the business component and makes it active.
2.b	Executes search on business component in the Save Bus Comp field. This locates a record ID that becomes active in the business component.

Table 6. Description of Labels in Figure 2

Label	Description
3	Performs text substitution on the question text before it displays. Any parameters in the Auto Sub Parm field that also appear in the question text convert to the values specified by the parameters.
4	After a user answers a question, if the Save Bus Comp field has a value, the active record saves the user's answer.
5	If the Save User Parameter field references a parameter, the answer is saved to that parameter.
6	All question leave events execute. For example, Question_Preleave, Question_Prebranch, and so on.

5 Upgrading SmartScripts

This chapter contains an overview of the tasks required to migrate a pre-Siebel 7 SmartScript to the current release. Each topic contains a cross-reference to detailed instructions within this guide. This chapter also describes how to upgrade Script Wizards to SmartScripts. It also outlines performance and design implications you need to consider before you migrate data from the SmartScripts created in the previous releases to the current release. It covers the following topics:

NOTE: As a general rule, always unrelease the released version of a script, and release it again after the upgrade process.

Overview of Migrating from Siebel 7 or Earlier Releases of SmartScript

The following section provides an overview of the issues involved in migrating data from Siebel 7 or earlier releases of SmartScript. Cross-references point you toward other chapters in this guide that provide detailed instructions:

- **VB and eScripts.** Support any VB or eScript that has been written inside Siebel SmartScript after the upgrade, because it runs on the server. Any upgrade issues with upgrading the code are generic to the application and are discussed in the *Siebel Database Upgrade Guide*. For information on VB and eScripts see Chapter 8, "Extending Scripts with Siebel VB and Siebel eScript."
- **Question displays.** Post-Siebel 7 SmartScript questions display differently than in earlier releases. With version 7 and later releases, the maximum number of questions in a given page display at once in a *page section*. Question Reveal functions are 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. For more information, see Chapter 4, "Working with Questions, Answers, and Translations."
- **Question translations.** Every question in your SmartScript must have a translation in the same language in order for the SmartScript to execute.
- **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 6, "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 pre-Siebel 7 SmartScript is no longer supported. See Chapter 6, "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 the current time elapsed is only available to customers using Siebel SmartScript along with CTI. See Chapter 10, "Modifying the Customer Dashboard," for more information.
- **Script Wizard.** Post-Siebel 7 SmartScript includes a feature to convert the Script Wizard scripts to SmartScripts. Script Wizard is no longer supported as a separate feature. See "Converting a Script Wizard into a SmartScript" on page 48 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 Script Wizard view of the Administration - SmartScript screen.

To convert a Script Wizard into a SmartScript script

1. Navigate to Administration - SmartScript screen > Script Wizard view.
2. Select or import the Script Wizard that you want to convert.

   You can import Script Wizards from a different environment but all Script Wizards must be upgraded before they can run in a post-Siebel 7 environment.
3. Select Convert from the Script Wizards menu.

   The confirmation view appears. After you click OK, the script no longer appears in the Script Wizard view, but only in the SmartScript view.
4. Navigate to the Administration - SmartScripts screen > Scripts view.
5. Select the script that you converted in Step 2. Make sure that the value that appears in the Business Component field is the name of the business component on which the script button that invokes the SmartScript is based.

   For more information, see "Creating Questions" on page 32.

   NOTE: The Business Component column does not appear by default in the Scripts view.

6 Navigate to the Administration - SmartScripts screen > Questions view, and select the first question in the script, then enter a value in the Save Business Object field.

For more information, see "Creating Questions" on page 32.

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

If the answer data is to be selected from an applet, the Save Field must specify the appropriate data for that applet, and a Pick Applet or Mvg Applet must 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 the other details in the script to make sure that no part of the script 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 allows the new SmartScript to automatically execute from the script buttons on the Account, Opportunity, and Contact profile views.

6 Customizing a SmartScript User Interface

This chapter describes how to customize the user interface of a SmartScript. It includes the following topics:

About Customizing a SmartScript User Interface

Siebel SmartScript provides a flexible user interface that allows modification. The answer controls that determine whether an answer appears as a text box, drop-down list, or pick applet are determined by values in the question definition. See Chapter 3, "Fundamentals of Creating SmartScripts," for more information on working with questions.

Two other important ways in which the SmartScript user interface is designed are listed as follows:

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

Determining which questions you want to 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, Siebel SmartScript 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 might 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.

Siebel SmartScript allows you to apply and modify HTML formatting code to determine the way question and page data displays. You can work with HTML formatting 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 the other design criteria that you are considering, because of the mechanisms involved in a question's definition. Siebel SmartScript processes each question, and if the processing of a question requires certain events, then it might 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 there must be a page break between these questions. Or, a question might have code executed when the question is completed to check some values in the answer before moving to the next question.

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

- If a question has a VB leave event, such as Question_PreBranch, Question_PreLeave, or Question_Leave, Siebel SmartScript always breaks a page after that question.
- If a question has a VB enter event such as Question_Enter or has a Search Spec, the page always breaks 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), Siebel SmartScript breaks a page before that question.
- If a question is on a Save Bus Comp and that business component is not positioned on any record, SmartScript breaks the page before that question, unless the first question in that page is also on the same business component. This rule allows new records to be created.

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

NOTE: You can manipulate Siebel 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 causes any other effect.

About HTML in the SmartScript User Interface

In SmartScript, 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 basic 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:

- 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.

For more information, see the following topics:

- "Formatting Question Text Using HTML Tags" on page 53
- "Adding Images" on page 54
- "Adding URLs" on page 54
- "Using HTML to Modify the Design Template" on page 54

Formatting Question Text Using HTML Tags

Using standard HTML tags in the question text, you can format any text that displays 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.

The following example renders the text in bold:

```
<B>Be sure to enter 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 Siebel SmartScript, including making the text of Information questions bold, 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 must provide an explicit path to an image file like a GIF or JPEG, and this image displays when that question translation text is rendered inside the SmartScript player applet.

Example:

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

Adding URLs

You can add URLs to questions to provide links to static, non-SWE-generated HTML pages, by adding a URL reference tag inside the question translation text. Always open these URLs in a separate browser. The URLs must not reference Siebel pages.

Example:

```
You can go to the<a href="http://www.oracle.com" target="_blank">Oracle 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 want to use such elements in a SmartScript.

Using HTML to Modify the Design Template

SmartScript pages are dynamically generated HTML content that combine 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 values from the SmartScript definition.

Typically, you 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 first to last in a single column. Each question starts directly after 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 can 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.

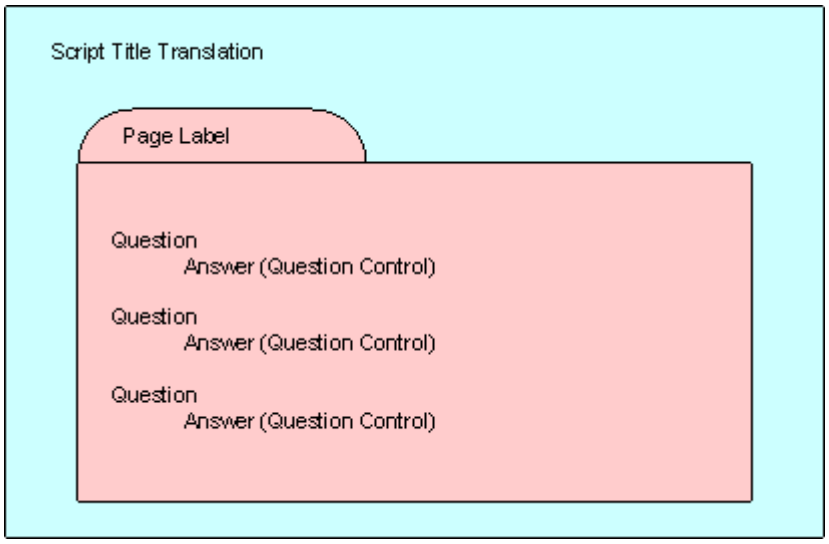

Figure 3. The Template Refers to This Basic Page Layout Design

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

You can make some changes to the SmartScript view using Siebel Tools. You can modify the view on which the SmartScript player applet appears to add or remove other applets. For example, you can remove the tree control in the employee SmartScript views 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 might occur, such as data not displaying correctly.

For more information on using Siebel Tools, see *Configuring Siebel Business Applications* and *Siebel Developer's Reference*.

Customizing the Revenue Schedule SmartScript

This topic describes what you can and cannot customize in the Revenue Schedule SmartScript. The Revenue Schedule SmartScript generates revenues associated with an opportunity. One execution of this SmartScript can create multiple revenue records, depending on the answers to certain questions.

You use the Administration - SmartScript screen to customize this SmartScript. The following list describes the changes that you can make to this SmartScript:

- The order in which questions display.
- How questions display.

  This applies to properties of the question such as style, color, and the grouping of questions on a page.
- The business object, business component, and field that a question answer is saved to.
- Questions to ask.

  Users can create only new questions that directly set revenue fields based on the answers. No special logic can be built on these new questions.

The following list describes the changes that you cannot make to the Revenue Schedule SmartScript:

- Predefault values for questions.
- Logic that handles the revenue creation process.

  For example, you cannot modify questions, such as Number of new revenue periods and Replace existing revenues nor the logic that divides Total revenue into Periodic revenues.

If you want to extend or modify the functionality of the Revenue Schedule SmartScript, you can write Siebel VB or Siebel eScript code to implement the required functionality. This code is then invoked when you execute the Revenue Schedule SmartScript. For more information, see Chapter 8, "Extending Scripts with Siebel VB and Siebel eScript."

7 Verifying, Testing, and Invoking SmartScripts

This chapter includes information on verifying, testing, and invoking SmartScripts. It covers the following topics:

About the Verification Wizard

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

- Pages that cannot be reached by branching
- Questions that cannot be reached by branching
- Missing answer branches
- Missing translations

The Verification Wizard also attempts to clean up dangling references and remove unreachable branches. These problems might occur when you delete elements of an existing SmartScript. A dangling reference or an unreachable branch can cause a SmartScript not to execute or export. If you select the check boxes to clean up dangling references and remove unreachable branches, the Verification Wizard removes these elements from the SmartScript. Do not select these options if you have not finished editing the SmartScript and plan on making revisions. Otherwise, the SmartScript elements with which you are still working might be deleted.

To verify a SmartScript

1. Navigate to the Scripts view in the Administration - SmartScript screen, and select a SmartScript.
2. Select Verify from the Menu button.

   The Wizard starts and displays the Verify Intro view.
3. Select the appropriate check boxes to make your selections, then click Next.

   The Wizard checks the page branches and displays a list of any pages that cannot be reached.
4. Click Next for the next three screens to display:
   - Pages that are not referenced properly in the SmartScript
   - Pages that are not reachable by branching
   - Other missing attributes

5 Click Next.

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

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

7 Verify your SmartScript after you make corrections in case you miss errors or introduce new ones.

8 When the Verification Wizard does not return any errors, execute your SmartScript to test different answers at any branch points to make sure that it executes as you intend.

About SmartScript Diagnostics

You can enable SmartScript diagnostics using event logging. To turn on SmartScript logging, set the event log level for GenericLog to 3 or 4. For information on how to set log levels, see the *Siebel System Monitoring and Diagnostics Guide*.

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

The following topics describe different ways of invoking SmartScripts:

■ “Invoking SmartScripts Using a Hyperlink” on page 65

Invoking SmartScripts Using the User Interface

An employee can invoke a SmartScript from the user interface of the SmartScripts list view.

To invoke a SmartScript from the SmartScript user view

1. Navigate to the SmartScripts screen > SmartScripts view.

   The list of available SmartScripts appears.

2. Select the SmartScript that you want to invoke, then click the SmartScript name.

   This invokes the SmartScript that you selected.

Setting SmartScripts to Open Automatically

SmartScripts can run automatically when a user navigates to a particular view. To configure SmartScripts to run automatically, add the SmartScript Player applet to the view, and associate a SmartScript with the view. Use Siebel Tools to add the SmartScript Player applet to a view. See *Using Siebel Tools* for information. The following procedure describes how to associate a SmartScript with a view.

To set SmartScripts to open automatically

1. Navigate to Administration - SmartScript screen > Views view.

   The list of available views appears.

2. Select the view that activates a SmartScript when this view is accessed.

3. In the Auto SmartScript field, select the SmartScript to run when the view selected in Step 2 is accessed.

Invoking SmartScripts Using Siebel CTI

You can configure Siebel CTI to invoke a SmartScript on an agent's Siebel application. For more information, see *Siebel Communications Server Administration Guide*.

CTI parameters (from the telephone switch) can be processed and used within a SmartScript. These can be accessed using 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 Using Siebel VB or Siebel eScript

You can invoke SmartScripts programmatically using the application invoke method RunSmartScript or the applet invoke method RunCallScript. The invoke method, RunCallScript, invokes a SmartScript from an applet so the SmartScript updates 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, appletName

Argument	Description
callScriptName	Name of the SmartScript, or “” (quotation marks) if pathId is specified
pathId	Row ID of the SmartScript, or “” if ScriptName is specified
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
appletName	Specify the name of the applet if you want to invoke the SmartScript on a view where the applet is a clone of the SmartScript Player Applet

Returns

Not applicable

Usage

For consistency, the arguments to RunSmartScript are identical to those of RunCallScript, except for the additional *viewName* and *appletName* parameters. The *viewName* 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.

Siebel VB Example

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

Siebel eScript Example

```
InvokeMethod (“RunSmartScript”, “Pentium II Script”, “”, “ENU”, “USD");
```

Siebel Applet Button Example

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

RunCallScript

This function is similar to RunSmartScript in that it programmatically invokes a SmartScript. However, RunCallScript positions the SmartScript on the currently active record from the applet from which the SmartScript 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 using VB in order to pass in the additional parameters.

Syntax

RunCallScript SmartScriptName, 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 update an existing record. Use RunCallScript behind a button on an applet. This usage works in an identical manner to the script button on the Account, Contact, and Opportunity Profile views.

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

This topic describes the options available to you to cancel, finish, or resume a SmartScript. You can invoke three methods on a SmartScript object, using either Siebel eScript or Siebel VB. These methods are:

- CancelSmartScript

- FinishSmartScript
- ResumeSmartScript

You invoke these methods on the SmartScript object that you have currently running. The CancelSmartScript and FinishSmartScript do not require parameters. The syntax for the ResumeSmartScript method is as follows:

```
ResumeSmartScript viewName, appletName
```

The following table describes the parameters that ResumeSmartScript accepts.

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

In addition to the methods listed previously, an applet user property exists that defines the behavior when you navigate to a different view while a SmartScript is currently running. This applet user property is AutoViewMode. Table 7 describes the values it accepts and the resulting behavior.

To add the AutoViewMode applet user property to an applet

1. Log in to Siebel Tools.
2. In the Object List Editor, locate the applet to which you want to add the AutoViewMode applet user property.
3. Create a new record in the Applet User Properties list with the name AutoViewMode and a value as described in Table 7.

Table 7. Parameter Values for AutoViewMode

Value	Behavior
Resume	Always resume the currently running SmartScript (default setting) if the user returns to the SmartScript.
Cancel	Cancels the currently running SmartScript if the SmartScript associated with the view is different from the currently running SmartScript. Otherwise, the processing of the SmartScript resumes.
Save	Saves the currently running SmartScript (so the user can review and resume it later from my saved session) if the SmartScript associated with the view is different from the currently running SmartScript. Otherwise, the processing of the SmartScript resumes. **NOTE:** In order to resume a SmartScript at a later date, the SmartScript must be released before it is saved. If it is not released, it will not be possible to resume the SmartScript.

Invoking a SmartScript from a Currently Running SmartScript

You can invoke a SmartScript from another SmartScript using Siebel VB or Siebel eScript. Cancel or finish the currently running SmartScript, and then use the RunSmartScript invoke method to start a new SmartScript.

Usage

You can create a SmartScript that makes 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 Using a Hyperlink

This topic describes how to invoke a SmartScript from a hyperlink. You can do this from an email or from a Web page, generated by an application, such as Siebel Self Service. The following example demonstrates how to invoke a SmartScript from a custom view in the Siebel Self Service application. To use this functionality, you must modify objects in local.

To invoke a SmartScript from a custom view in Siebel Self Service

1. Log in to Siebel Tools.
2. Query for the applet from which the SmartScript is invoked. In the example, this applet is Self Service Account Quick Links Applet.
3. Lock the project that contains this applet.
4. Create two new controls for the Self Service Account Quick Links Applet. In the example, these controls are LinkSmartScript and LabelSmartScript. The following table describes the property values for the LinkSmartScript control.

   NOTE: In the following tables, X indicates that you select a check box.

Property	Value
Name	LinkSmartScript
Caption	SmartScript
HTML Display Mode	EncodeData
HTML Row Sensitive	X
HTML Type	Text
Method Invoked	GotoView
Sort	X
Text Alignment	Left
Visible	X
Visible - Language Override	Y

The following table describes the property values for the LabelSmartScript control.

Property	Value
Name	LabelSmartScript
Caption	Create a SmartScript
HTML Display Mode	EncodeData
HTML Row Sensitive	X
HTML Type	Label

Property	Value
Sort	X
Text Alignment	Left
Visible	X
Visible - Language Override	Y

5 Invoke the Edit Web Layout command for the Self Service Account Quick Links Applet and add the LinkSmartScript and LabelSmartScript controls to the applet.

6 For the LinkSmartScript control, add a Control User Prop with the values listed in the following table.

Property	Value
Name	View
Value	myExample SmartScript Player View **NOTE:** You create this view in Step 7.

7 Navigate to the View object, and select the Smart Script Player View (eApps) view object. Lock the project, create a new view by copying this object, and rename it. This is the view that you specify for Value in the Control User Prop created in Step 6. The following table specifies the values to give to this newly created view object.

Property	Value
Name	myExample SmartScript Player View
Project	eSmartScript
Business Object	Smart Script Player

8 Navigate to the Screen objects, and select the Smart Script Player View (eApps) screen object.

9 Add the view that you created in Step 7 to the list of screen views for the Smart Script Player View (eApps) screen object.

CAUTION: Select a sequence number that does not conflict with existing sequence numbers.

10 Navigate to the Application objects, select the Siebel Self Service application, then navigate to the Screen Menu Item list, and make sure that the Smart Script Player View (eApps) screen appears in this list.

11 Compile the locked projects.

12 Log in to your Siebel application, and navigate to the Administration - Application screen > Views view.

13 Create a new record in the Views list, add the view that you created in Step 7, and select the Default Local Access check box.

14 In the Responsibilities list, add the responsibilities that you want to allow access this view.

15 Navigate to the Administration - SmartScript screen > Views view, select the view you created in Step 7, and in the Auto SmartScript field select the SmartScript that you want to invoke.

16 Log out from your Siebel application.

17 Restart your Siebel Server.

After the Siebel Server restarts, you can log in to Siebel Self Service where a hyperlink entitled, Create a SmartScript appears. Clicking this hyperlink invokes the SmartScript that you specified in Step 15.

8 Extending Scripts with Siebel VB and Siebel eScript

This chapter describes how you can extend the functionality of your SmartScripts using Siebel eScript or Siebel Visual Basic (VB). It includes the following topics:

About Siebel VB and Siebel eScript

You can use Siebel VB and Siebel eScript (a scripting language like JavaScript) with all SmartScript elements. This topic describes the Siebel VB and Siebel eScript events that are specific to SmartScripts. The syntax for all of these events is the same in both Siebel VB and Siebel eScript. Most of the examples in this chapter are in Siebel VB. However, the events work the same way in Siebel eScript. For more information about Siebel VB and Siebel eScript, see the following guides:

- *Siebel Object Interfaces Reference*
- *Siebel VB Language Reference*
- *Siebel eScript Language Reference*

NOTE: Siebel eScript functions work in the same way as the standard JavaScript functions. They require the trailing parentheses ( ) even when the function does not have any 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 Siebel VB and eScript events, 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. Table 8 lists the standard Siebel VB and Siebel eScript events that are used in SmartScript events.

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

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

Each question element is configured with four events. Other elements do not have Siebel VB or Siebel eScript events associated with them. These events can contain methods that can be used to further control the workflow of a script. Table 9 lists the standard Siebel VB and Siebel eScript events that are used in SmartScript Questions.

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

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

About SmartScript Object Types

Within a script, you can refer to virtual business components in the same way that you refer to other business components. Virtual business components are a class of business component that access external data (data stored outside the Siebel database). Virtual business components allow you to present and manipulate external data using 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

Accessing the Scripting Area

This topic describes how you access the scripting area in the Administration - SmartScript screen. The view that you access in this screen depends on whether you want to add script to a script or to a question.

To access the scripting area

1. Navigate to the Administration - SmartScript screen.
2. Select the option from the following table that applies to you.

If you want to ...	Then ...
Add scripting to a SmartScript object	Navigate to the Scripts view > Programs subview
Add scripting to a SmartScript Question object	Navigate to the Questions view > Programs subview

3. Create a new record in the Programs view, select the event type from the Name drop-down list, and the Script language from the Program Language drop-down list.
4. Click Save in the Programs view.
5. Add your script in the Scripts text area that appears after the Programs subview.

   NOTE: Each script has the option of using the Siebel VB or Siebel eScript languages. However, you must use the same language throughout a script. For a question, you also have the option of using Siebel VB or Siebel eScript. Every question method has to use the same language for each question. This means that the languages can be different for different questions, but within the same question, the languages must be the same.

SmartScript Events

You can configure how the SmartScripts respond to an event, using the Programs subview of the Administration - SmartScript screen > Scripts view.

The following topics describe the available events:

Script_Open

Script_Open allows updating of the Customer Dashboard. You can also position the database on the appropriate record when saving the answers to a business component.

Syntax

Script_Open

Returns

Not applicable

Usage

Script_Open allows you to set the variables in complex branching, in later questions, or in the Customer Dashboard. You can also use it to create a new record and populate some values using code.

Script_Cancel

Script_Cancel handles postprocessing when a script is canceled.

Syntax

Script_Cancel

Returns

Not applicable

Usage

This postevent is called to allow any last-minute cleanup or postprocessing when the script is canceled. Script_Cancel is not called if Cancel is called from the Script_Open event. If an error is raised during the Script_Cancel procedure, the error is 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 wants to finish the SmartScript (by clicking the Finish button). This is a good place to check additional constraints on the SmartScript 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 postevent that is called to permit any last-minute cleanup or postprocessing 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.

SmartScript Methods

You can use the following SmartScript methods with any of the events that you access from the Programs subview of the Administration - SmartScript screen > Scripts view. For more information about these events, see “SmartScript Events” on page 72. The available methods are:

Cancel

Cancel cancels the current script.

Syntax

Cancel

Returns

Not applicable

Usage

The Cancel method stops the SmartScript's execution. This method cancels only 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 method always returns a question object if the script is actively executing. The 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 zero (0) if the SmartScript object has not been specified to execute (used when listing available SmartScripts).

Finish

Finish causes the current script to finish.

Syntax
Finish

Returns
Not applicable

Usage
The Finish method causes the currently running script to finish. 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 created when a script is launched 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 77, GetContactId on page 78, and SetCampaignId on page 83.

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 77, GetContactId on page 78, and SetCampContactId on page 83.

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 77, GetCampContactId on page 77, and SetContactId on page 84.

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 nontranslated name set during authoring, not the label of the page tab displayed for a particular language.

Siebel VB Example

```
Dim IntroductionPage as SmartScriptPage

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

See Also

CurrentPage on page 75 and Page on page 103.

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

You use GetParameter 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.

You use GetParameter 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 using the Siebel Communications Server. For more information, see *Siebel Communications Server Administration Guide*.

You use this parameter 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 using Siebel Communications Server.
CampConId	Set when SmartScript is invoked using Siebel Communications Server.
ContactId	Set when SmartScript is invoked using Siebel Communications Server.
ScriptId	Script row ID.
ScriptName	Script name.
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 records are 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 85.

GetQuestion

GetQuestion returns a question of the script by name.

Syntax

```
varPage.GetQuestion(name)
```

where varPage is the Page object.

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

Returns

A question of the script as a SmartScriptQuestion.

Usage

Specify the name of the question as a parameter to the GetQuestion method to retrieve the question as a SmartScriptQuestion. The name you specify is the nontranslated name set during authoring, not the question text displayed for a particular language. GetQuestion is declared as a SmartScriptQuestion.

You must also specify the page where the question appears. If the question appears on the current page, you do this as follows:

```
var Question = Page().GetQuestion ("Name of question");
```

If the question appears on another page, you must retrieve this page as in the following example:

```
var p=Script().GetPage("Other page name");

var Question = p.GetQuestion ("Name of question");
```

Siebel VB Example

```
Dim ContactQuestion as SmartScriptQuestion

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

See Also

GetQuestionEnable on page 99.

GetSessionId

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

Syntax
GetSessionId

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

Usage
GetSessionId is declared as a string. Call this method from either Script_Finish or Script_Save events.

Siebel VB Example
```
Dim ScriptSessionId as String

ScriptSessionId = GetSessionId()
```

Siebel eScript Example
```
Var ScriptSessionId;

ScriptSessionId = GetSessionId();
```

NOTE: In Siebel eScript make sure to include the Header and Footer "function Script_Finish() {" and "}" or else the script executes without returning a value from GetSessionId().

OriginalDashboardText

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

Syntax
OriginalDashboardText

Returns
The text of the Customer Dashboard that was 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 obtained using 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 obtained using script execution. This information is stored in the SmartScript session.

See Also

GetCampaignId on page 77, SetCampContactId on page 83, and SetContactId on page 84.

SetCampContactId

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

See Also

GetCampContactId on page 77 and SetContactId on page 84.

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 obtained using script execution. This information is stored in the SmartScript session.

See Also

GetContactId on page 78, SetCampaignId on page 83, and SetCampContactId on page 83.

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

Use SetUserParameter to assign the value returned by 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 79.

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 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?" is rendered as "Are you telephoning from your carriage?" if the function was configured to translate from U.S. 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 do not affect the displayed values in the Customer Dashboard. For more information on how to affect the customer dashboard values., see Chapter 10, "Modifying the Customer Dashboard."

SmartScript Page Methods

The SmartScript Page object type represents a single page of the SmartScript being executed.

You can invoke the following methods on the SmartScript Page object:

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 can be used as reference help text that 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 text is displayed 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 nontranslated name set during authoring. It is 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.

SmartScript Question Events

You can configure how SmartScript Questions respond to certain events by writing code in the Programs subview of the Administration - SmartScript screen > Question view. The available events are:

- “Question_Enter” on page 90

Question_Enter

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

Syntax

Question_Enter

Returns

Not applicable

Usage

This postevent 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 preevent called by certain methods before the user leaves a question.

Syntax

Question_PreLeave

Returns

Not applicable

Usage

This preevent 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 answer given to the question as a string.

Returns

An evaluation of an answer, which is 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 with 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 answer that is stored. The value returned in the parameter is not stored as the answer to the question, but is used to choose the answer used for branching.

For example, if the caller supplies a bank account number, the function evaluates the number to determine what type of account it is and branches to questions for that type of account. The answer stored in the database, however, is the 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 the 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
```

SmartScript Question Methods

You can invoke the following methods on the SmartScript Question object:

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. You can use this method 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 might have changed many times as the user worked through the script. GetCurrentCurrencyCode is declared as a string.

See Also
GetInitialCurrencyCode on page 96 and GetPriorCurrencyCode on page 97.

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 it is a currency question). This might have changed many times as the user worked through the script. GetCurrentExchangeDate is declared as a string.

See Also

GetInitialExchangeDate on page 96 and GetPriorExchangeDate on page 98.

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 97, GetPriorValue on page 98, and SetCurrentValue on page 105.

GetHelpText

If any help text is present, 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 can 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 null 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 94 and GetPriorCurrencyCode on page 97.

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 null 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 94 and GetPriorExchangeDate on page 98.

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 null 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 95 and GetPriorValue on page 98.

GetPriorCurrencyCode

GetPriorCurrencyCode returns the currency code entered the previous time that 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 the previous time. The value returned 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 94 and GetInitialCurrencyCode on page 96.

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 it is a currency question) before the user reached it most recently. The value returned 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 94 and GetInitialExchangeDate on page 96.

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. The value returned 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 95 and GetInitialValue on page 97.

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 86, OriginalQuestionText on page 102, and SetQuestionEnable on page 106.

GetSaveBusComp

GetSaveBusComp returns the business component that was configured to store the answer.

Syntax

GetSaveBusComp

Returns

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

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 NULL. 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 103.

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 104.

HasDefaultAnswer

HasDefaultAnswer returns a Boolean value indicating whether the question has a default answer.

Syntax

HasDefaultAnswer

Returns

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

Usage

HasDefaultAnswer is declared as Boolean.

MustAnswer

MustAnswer returns a value indicating whether 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 it is 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 value displayed is obtained using 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 99 and SetQuestionEnable on page 106.

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 79.

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 that stores the answer 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 using SaveFieldName.

See Also

GetSaveBusComp on page 100, SaveBusObjName on page 104, and SaveFieldName on page 104.

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 that stores the answer. This must be specified if a business component and field are specified using SaveBusCompName and SaveFieldName, respectively. It is declared as a string.

See Also

GetSaveBusObj on page 101, SaveBusCompName on page 103, and SaveFieldName on page 104.

SaveFieldName

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

Syntax

SaveFieldName

Returns

The name of the field that stores the answer 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, because not all questions store their answers in a field. (Some questions do not save answers.) It is declared as a string.

See Also

SaveBusCompName on page 103 and SaveBusObjName on page 104.

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 if the user had entered it. All validation and branching are activated. If the question is a currency question (that is, the answer type is *Currency*), specify the currency code and exchange date 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, do not use SetCurrentValue() 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 95.

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 99 and SubstituteText on page 107.

SubstituteText

See “SmartScript Methods” for a description and a VB example of this method.

See Also

GetQuestionEnable on page 99 and SetQuestionEnable on page 106.

WasAnswered

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

Syntax

WasAnswered

Returns

TRUE if the user answered the question, or 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 return TRUE even if the agent merely passes through them.

WasAnswered is declared as type Boolean.

Improving the Performance of SmartScripts

You can improve the performance of your SmartScripts by setting the value of the business component user property, DeactivateBCField, to TRUE. This disables all business component fields except those used in SmartScript Questions. The default value for DeactivateBCField is FALSE.

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

If you set DeactivateBCField to TRUE and you want to work with a field that is not in your SmartScript Question, you need to call ActivateField before you can work with the required field.

Invoking a Business Service from a SmartScript

You can invoke a business service from a SmartScript by using Siebel VB or Siebel eScript to make call statements for the required business service. This allows you to reuse code. The following example code retrieves information from a SmartScript and invokes the Universal Inbox business service to create a new item in the Inbox. For another example, see “Invoking Siebel Assignment Manager” on page 111.

```
function Script_Finish ()

{

   var szScriptSessionId;

   var szRequester;

   var svc;

   var indata;

   var outdata;

   var ssPage;

   var ssQuestion;

   var EmployeeLastName;

   var EmployeeFirstName;
```

```
    var RequestedChange;
    var PAFComments;
    var PAFPriority;
    var szSSLanguageCode;
    var szSSCurrencyCode;
ssPage = GetPage("PAF Which Change");
ssQuestion = ssPage.GetQuestion("Display Employee Last Name");
EmployeeLastName = ssQuestion.GetCurrentValue();
ssQuestion = ssPage.GetQuestion("Display Employee First Name");
EmployeeFirstName = ssQuestion.GetCurrentValue();
ssQuestion = ssPage.GetQuestion("PAF Change Requested");
RequestedChange = ssQuestion.GetCurrentValue();
ssQuestion = ssPage.GetQuestion("PAF Comments");
PAFComments = ssQuestion.GetCurrentValue();
ssQuestion = ssPage.GetQuestion("PAF Priority");
PAFPriority = ssQuestion.GetCurrentValue();
    // Cancel saving everything to the database
    Cancel ();
indata =TheApplication ().NewPropertySet ();
outdata = TheApplication ().NewPropertySet ();
    // Get the login name of the user
    szRequester = TheApplication ().LoginName ();
// Get SmartScript Save Session table Id.
szScriptSessionId = GetSessionId ();
    szSSLanguageCode = GetParameter("Language");
    szSSCurrencyCode = GetParameter("Currency");
indata.SetProperty ("SmartScriptLanguageCode", szSSLanguageCode);

// ItemObjectId, ItemType, ItemSubmittedBy, and ItemDescription are the
// required input arguments for the "Universal Inbox.Initialize"
```

```
indata.SetProperty ("ItemObjectId", szScriptSessionId);

// Item Type is the Approvals Inbox type defined in the

// Approvals Inbox Administration screen

indata.SetProperty ("ItemType", "Personnel Action Form");

// Short Description of the inbox item

indata.SetProperty ("ItemDescription", RequestedChange + " PAF" + " for " +
EmployeeFirstName + " " + EmployeeLastName);

indata.SetProperty ("ItemSubmittedBy", szRequester);


// ItemQueueDuration, ItemPriority, and ItemComments are the

// optional input arguments for the "Universal Inbox.Initialize"

indata.SetProperty ("ItemQueueDuration", "129600");

indata.SetProperty ("ItemPriority", PAFPriority);

indata.SetProperty ("ItemComments", PAFComments);

svc = TheApplication ().GetService ("Universal Inbox");

svc.InvokeMethod("Initialize", indata, outdata);


var id = GetSessionId ();

var bo = TheApplication().GetBusObject("Smart Scripts");

var bc = bo.GetBusComp("Call Script Runs");

bc.ActivateField("Status Code");

bc.SetSearchSpec("Id", id);

bc.ExecuteQuery();

if (bc.FirstRecord())

   bc.SetFieldValue("Status Code", "Finished");

bc.WriteRecord();

}
```

Invoking Siebel Assignment Manager

You can invoke Siebel Assignment Manager from SmartScripts using Siebel eScript or Siebel VB. How you invoke Assignment Manager depends on the mode in which Assignment Manager is operating. If Assignment Manager is executing in interactive mode, you require a user (for example, a service representative) to perform some action (for example, assign a record to an employee) to advance the script.

If Assignment Manager is executing in dynamic mode, you can invoke it from a SmartScript by triggering certain events. A user is not required to invoke the Assignment Manager from the SmartScript.

The following example procedure demonstrates how to invoke Assignment Manager executing in dynamic mode on a newly created service request record.

To invoke Assignment Manager

NOTE: This procedure is an example of how you can invoke Assignment Manager from a SmartScript.

1. Create assignment rules for the service request object.

   For more information about assignment rules configuration, see *Siebel Assignment Manager Administration Guide*.

2. Create a SmartScript with values similar to those in the following table.

Element	Attribute	Value
Question1	Name	SRDescription
	Answer Type	String
	Answer Control	Default
	Save Business Object	Service Request
	Save Business Component	Service Request
	Save Field	Description
	Translations	English-American
Question 2	Name	TestQuestion2
	Answer Type	RDsr2
	Answer Control	RDsr2
	Translations	English-American

Element	Attribute	Value
Page	Name	TestPage1
	First Question	SRDescription
	Translation	English-American
	Label	TestPage1
	Next Question	TestQuestion2
SmartScript	Name	SmartScript_AssignSR
	Type	Pick any value (Other)
	Active	Yes
	First Page	TestPage1
	Translation	English-American
	Label	SmartScript Assign SR

3 Navigate to Administration - SmartScripts screen > Scripts view > Programs subview.

4 In the Programs list, select Script_Finish, and add the following code:

NOTE: When assigning an object, it must exist in order to match the assignment rules, criteria, and so on. To make sure that a newly created record exists in the database, users can run the following event:
`BusinessComponent.WriteRecord()`
before calling an Assignment Manager method in the SmartScript event.

```
function Script_Finish()

{

var sFirstQuestion = StartQuestion();

var SaveBC = sFirstQuestion.GetSaveBusComp();

TheApplication().Trace(" \n the value of sFirstQuestion = "+sFirstQuestion);

SaveBC.ActivateField("Area");

SaveBC.SetFieldValue("Area", "Upgrade");

//*****************

var myid = SaveBC.GetFieldValue("Id");

SaveBC.WriteRecord();

var bsAmgr = TheApplication().GetService("Synchronous

Assignment Manager Requests");

var psInput = TheApplication().NewPropertySet();
```

```
var psOutput= TheApplication().NewPropertySet();

psInput.SetProperty("AsgnObjName","Service Request");

//Assignment Object Name

psInput.SetProperty("ObjRowId", myid);  //Object Row ID

bsAmgr.InvokeMethod("Assign", psInput, psOutput);

//*****************

}
```

Siebel VB and Siebel eScript Sample Code

The three samples in this section are designed to work as a unit but are separated for simplicity. The samples are:

- “Sample Code of Dynamic Questions” on page 113
- “Sample Code of Finding a Contact” on page 114
- “Sample Code of Complex Branching” on page 116

Sample Code of Dynamic Questions

The following sample code implements inserting an answer into a question dynamically. 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);

}
```

Sample Code of 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);

    }

}
```

Sample Code of 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. For sample code that demonstrates data acquisition and database querying, see "Sample Code of Finding a Contact" on page 114.

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);

}
```

9 Importing, Exporting, and Deploying SmartScripts

This chapter describes how to import, export, and deploy SmartScripts. It covers the following topics:

Siebel SmartScript allows the exporting and importing of SmartScripts in a binary format between the development, testing, or production environments. A file that can be imported or exported is referred to as a Siebel SmartScript file or SSS file.

If you have a large number of SmartScripts to deploy to another environment, consider using the Application Deployment Manager. For more information, see "About SmartScripts and the Application Deployment Manager" on page 121.

Exporting SmartScripts

You can export SmartScripts using a menu option available in the Scripts view of the Administration - SmartScript screen. When you export a SmartScript, all the elements required for a SmartScript are copied to the destination file. When this file is imported into another Siebel database, the SmartScript must 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 SmartScript

1. Navigate to the Administration - SmartScript screen > Scripts view.
2. Select the SmartScript you want to export, and choose Export Script from the Menu button.

   The SmartScript Export Results view appears.
3. Click the hyperlink that displays the name of the SmartScript to export the SmartScript.

   The File Download dialog box appears.
4. Click Save to download the SmartScript.

   The Save As dialog box appears.
5. Specify a location to save the SmartScript, then click Save.

   The SmartScript is exported to the location that you specify.

Importing SmartScripts

You can import SmartScripts using a menu option, which is available in the Scripts view of the Administration - SmartScript screen. The import process checks many characteristics of the SmartScript 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, you can view a log file listing any errors 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 SmartScript, the SmartScript that you want to create or update is based on the SmartScript name for the exported SmartScript, not the file name of the SmartScript or the name of the selected SmartScript when the Import button is selected.

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

NOTE: For Siebel Innovation Pack 2015, the local database for the Siebel Mobile Web Client uses SAP SQL Anywhere. SAP SQL Anywhere is not available for new deployments after September 2015. For more detailed information on how this change affects Siebel Tools and Siebel Remote, see *Siebel Release Notes* on Siebel SupportWeb for Innovation Pack 2015 (Doc ID 1996273.1).

After a SmartScript has been imported, verify it by using the Verification Wizard, and then perform normal testing. For information about the Verification Wizard, see "About the Verification Wizard" on page 57.

To import a SmartScript

1. Navigate to Administration - SmartScript > Scripts.
2. Click Menu, and choose Import Script.

   The SmartScript Import Intro view appears.
3. In the File Name field, click the Select button to display the Add Attachment dialog box.
4. Navigate to where the SmartScript is stored, then select it.
5. From the In case of error drop-down list, select how you want to handle conflict resolution.

   For more information, see "Resolving Conflicts Encountered During an Import" on page 120.
6. Click Import File.

   The SmartScript is imported. A list of the different elements that were processed as part of the import appears in the Processing Steps list applet. The import process also generates a log file that you can view when you click the hyperlink that displays the name of the log file.

Resolving Conflicts Encountered During an Import

If Siebel SmartScript encounters conflicts during the import process, it refers to the options you select in the In case of error field. You can select from the following options: Update, Skip, or Add.

These options determine how Siebel 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 you choose the Add option, a duplicate version of the elements in the SmartScript is created (Script, Page, or Question). This duplicate version of the SmartScript has a different name. For example, if you import a SmartScript entitled Revenue Schedule Script with the Add option selected, the application creates a duplicate version of this SmartScript entitled, Revenue Schedule Script_1. The script references within the SmartScript continue to reference the original element name.

If Siebel SmartScript encounters any errors or conflicts, it creates a log file. You can view this log file when the import process has finished by clicking a link. Review the conflicts and errors reported in the log file, and correct each element as necessary, using the appropriate Administration - SmartScript view.

If the contents of a picklist are different in both the exporting and importing databases, a conflict occurs. Update the contents of the picklist before exporting the SmartScript, so that the contents are consistent in both databases.

About SmartScripts and the Application Deployment Manager

SmartScripts are one of the data types that the Application Deployment Manager (ADM) supports. Using ADM, you can import or export SmartScripts between different environments (for example, between development and production). For more information about Application Deployment Manager, see the *Siebel Application Deployment Manager Guide*.

10 Modifying the Customer Dashboard

This chapter provides an overview of the Customer Dashboard. It describes how you configure the SmartScript Player applet to invoke the Customer Dashboard and display it to the user. It includes the following topics:

For more information about modifying and configuring the Customer Dashboard, see *Configuring Siebel Business Applications*.

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 start, as long as the session is active and the Customer Dashboard is not closed. The Customer Dashboard can also be active when the Search Center or communications channels that Oracle support are active.

Example

The employee on an outbound telemarketing campaign logs in to a predictive dialer using the Siebel Communication toolbar. On logging in to 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, Siebel 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, the employee forgets key information about the customer or contact (for example, name, phone number, and so on), the employee can refer to the Customer Dashboard.

If a call center employee regularly needs additional information about the contact, the Customer 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 wants to search information, the Customer Dashboard can be customized so that the results of the search populate the Customer Dashboard. With the customization mentioned in the preceding paragraph, the employee can then navigate to information related to the search results.

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

Overview of Configuring the Customer Dashboard

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

The typical tasks to configure Customer Dashboard include:

- Designing the Customer Dashboard layout with Siebel Tools. The Customer Dashboard displays an applet based on a special Virtual Business Component (VBC).
- Linking business component fields to the Customer Dashboard fields, using Siebel Tools.
- Integrating the Communication Server with the Customer Dashboard. Communications Server administrators use the Communications Administration screen, which involves working with communications commands and events.

For more information about working with Siebel Tools, see *Using Siebel Tools*. 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 Upgrading to the Customer Dashboard

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 applet is configured to notify the Customer Dashboard. The default state of the SmartScript Player applet is TRUE, allowing for this notification. For more information, see “To access the SmartScript Player applet object” on page 125.
- Make sure that the updated parameters are configured in Siebel Tools to reflect the updated changes. For more information, see “To make sure the updated parameters are configured in Siebel Tools” on page 125.
- Update the Customer Dashboard by saving answers to the Customer Dashboard variables. For more information, see “To update to the Customer Dashboard” on page 124.

To update to the Customer Dashboard

1. Navigate to Administration - SmartScript screen > Questions view.
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 appears in the list of user properties for the Persistent Customer Dashboard business service in Siebel Tools.

   For example, “Fname”.
4. Navigate to the Scripts view > Translations list.

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 Customer Dashboard. Enter only parameters that have been configured in the dashboard business service in this field. You must also make sure that the SmartScript Player applet is configured to notify the Customer Dashboard. The default state is TRUE. For more information see *Configuring Siebel Business Applications*.

To access the SmartScript Player applet object

1 Lock your project in Siebel Tools.

2 In the Applets list, select Smart Script Player Applet (Tree Only).

3 Navigate to the Applet User Property list, and select Notify DashBoard.

4 Do one of the following:

- Set the value for Notify Dashboard to Y if you want the SmartScript Player applet to update the Customer Dashboard.

  This is the default value.

- Set Notify Dashboard to N if you do not want the SmartScript Player applet to update the Customer Dashboard.

To make sure the updated parameters are configured in Siebel Tools

1 Lock your project in Siebel Tools.

2 In the Business Service list, select Persistent Customer Dashboard.

3 Navigate to the Business Service User Prop list.

This list displays the current configuration of the Customer Dashboard.

4 Make sure that the parameters you entered in "To update to the Customer Dashboard" match the values in this list.

This list is specifically used for updating the Customer Dashboard from a SmartScript. You can also update the Customer 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 Deploy your changes to the Siebel runtime repository.

A SmartScript Tags

This appendix explains Oracle's Siebel SmartScript tag definitions by using an example. It covers the following topic: Script Tags for SmartScript.

Script Tags for SmartScript

The SmartScript tag definitions are explained using an example from 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 must 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

Q

T

U

V

W