Genesys®

Genesys Agent Scripting 8.1

User's Guide

About Genesys

Genesys is the world's leading provider of customer service and contact center software—with more than 4,000 customers in 80 countries. Drawing on its more than 20 years of customer service innovation and experience, Genesys is uniquely positioned to help companies bring their people, insights and customer channels together to effectively drive today's customer conversation. Genesys software directs more than 100 million interactions every day, maximizing the value of customer engagement and differentiating the experience by driving personalization and multi-channel customer service—and extending customer service across the enterprise to optimize processes and the performance of customer-facing employees. Go to www.genesys.com for more information.

Each product has its own documentation for online viewing at the Genesys Documentation website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Customer Care. Before contacting Customer Care, please refer to the ***Genesys Care Program Guide*** for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the ***Genesys Licensing Guide.***

Released by

Genesys Telecommunications Laboratories, Inc. www.genesys.com

Document Version: 81as_us_08-2014_v8.1.202.00

Genesys

Table of Contents

List of Procedures

Genesys

Preface

Welcome to the *Genesys Agent Scripting 8.1 Deployment Guide*. This guide describes system requirements, installation of the Genesys Agent Scripting application, installation of supporting applications, and Genesys Agent Scripting Collaboration mode.

This document is valid for the 8.1 release(s) of this product.

Note: For versions of this document created for other releases of this product, visit the Genesys Documentation website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at `orderman@genesys.com`.

This preface contains the following sections:

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on page 413.

About Genesys Agent Scripting

Genesys Agent Scripting is a customization solution available with Genesys Inbound Voice and Genesys Outbound Voice, a development tool that enables you to create and maintain browser-based web applications called “business process scripts.” A business process script guides agents through an effective customer interaction flow and helps them complete a structured business process.

Business process scripts provide many benefits. A script ensures the quality of the interaction, adds structure and clarity to the definition of the process, integrates data access and data entry capabilities, maximizes the probability of success, and minimizes the training required by the agent. Genesys Agent Scripting assists in constructing the interfaces to other back office systems.

Intended Audience

This guide is primarily intended for system administrators or script developers. It is also intended for contact center agents who will install and use Genesys Agent Scripting. The guide assumes that you have a basic understanding of:

- CTI concepts, processes, terminology, and applications.
- Network design and operation.
- Your own network configurations.
- You should also be familiar with:
  - Scripting using Active Server Pages (ASPs) or Java Server Pages (JSPs).
  - Apache Tomcat 4.1, 5.0, 5.5, 6.0; or Microsoft IIS 5.0, 6.0, 7.0 or 7.5 web servers.
  - HTML editing.

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to `Techpubs.webadmin@genesys.com`.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the scope of this document only and to the way in which the information is presented. Contact your Genesys Account Representative or Genesys Technical Support if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Contacting Genesys Technical Support

If you have purchased support directly from Genesys, please contact `Genesys Customer Care`.

Before contacting technical support, please refer to the *Genesys Customer Care Program Guide.*

Document Change History

This section lists topics that are new or that have changed significantly since the first release of this document.

GAS 8.1.2

In Table 57, “Target Environment Properties, General Tab,” on page 276, ” the Control variable name `Simulate URL` was changed to `Target URL`.

Chapter

1 Overview of the Development Environment

This chapter provides a brief overview of the Genesys Agent Scripting Development Environment.

The information in this chapter is divided among the following topics:

What is Genesys Agent Scripting?

Genesys Agent Scripting is a development tool that enables you to create and maintain browser-based web applications called “Agent Scripting .” An Agent Script guides contact center agents through an interaction and helps them complete a structured business process. Typical business processes include filling out an application, identifying the right type of expert contact to handle an inquiry, troubleshooting a system, and so on.

An Agent Script enables a contact center agent to complete an interaction in which the agent works with the contact to complete a task. The script helps the agent deliver a consistently high level of service to the contact.

In addition to providing a simple linear flow of script text and related controls, a Genesys agent script can react dynamically to information from a customer. This means that, using Genesys Agent Scripting, you can design an Agent Script to handle normal variations in a structured business process. As necessary, the script can respond dynamically to the answers provided by the customer. Depending on the feedback, the flow of the script may move from one course to another. For example, if the default flow is not correct for the contact's circumstances, the script may adjust by displaying a different page of text to the agent.

Agent Scripts provide many benefits. A script ensures the quality of the interaction, adds structure and clarity to the definition of the process, integrates data access and data entry capabilities, maximizes the probability of success, and minimizes the training required by the agent. Genesys Agent Scripting also assists in constructing the interfaces to other back office systems.

Genesys Agent Scripting creates web-based applications for any of the following platforms:

- Active Server Pages (ASP)
- Active Server Pages - Extended (ASPX)
- Java Server Pages (JSP)

Agent Scripting Hierarchical Structure

Agent Scripting uses a hierarchical structure that consists of the following elements:

```
Project Books
└──Projects
   └──Process Flows
      └──Streams (subprocesses)
         └──Pages
```

Genesys Agent Scripting stores the metadata (the information that describes the business process being created), in a Microsoft Access database (`.mdb` file) referred to as a *Project Book.*

Projects are high-level groupings of related business processes, for example, all processes associated with new customer acquisition.

A *Process Flow* represents a single business process from start to finish, such as signing up a new customer.

Streams are subprocesses that are likely to be reused, such as a credit card validation subprocess, or a subprocess that updates a database in a specific way.

Pages are the individual web pages that the script users (agents) see when the script application runs. Pages can contain text, Fields to display or obtain data, Action buttons that trigger execution of code, or Action code that is executed when a Page loads or unloads as the script user navigates through the Stream.

In addition, *Branching Logic* is used to map a path from one Page to another to create the sequence that script users navigate through. Branching can occur between any two Pages defined in the Project Book.

Database Interfaces provide easy access to back-end systems and their associated information. Data can be read from or written to the back-end system automatically. The data remains available throughout the interaction without any action on the part of the agent.

Organizing a New Project

When approaching a new project, it's important to think in terms of the individual *tasks* or *processes* that need to take place in order to complete an interaction or other business function.

Logical groups of closely-related tasks are the foundation on which we build a complete process or call flow. These tasks are commonly referred to as *work flows*, *process flows*, or in a contact center environment, *call flows*, *call types*, or *scripts*.

Genesys Agent Scripting uses specific organizational elements to capture Pages in logical groups to create a complete flow, or interaction. Refer back to "Agent Scripting Hierarchical Structure" on page 18 for a brief introduction to these organizational elements.

Streams, Process Flows, and Projects

Streams and Process Flows are methods of organizing Pages into a logical hierarchy:

- A Stream is a collection of Pages and their associated Branching Logic.
- A Process Flow is a collection of one or more Streams.
- A Project is a group of one or more Process Flows with a common organizing theme.

A Project Book is a collection of one or more Projects, and the highest level that can share common elements, such as Fields, Actions, and Pages.

Streams

Streams are used to group together Pages that perform a single logical function, such as a group of Pages that adds a new customer to a database, or that collects credit card information from a customer and validates the information. Streams can facilitate reuse of commonly-used functions. Once you've defined a Stream to collect and validate credit card information, you can use the Stream in any number of other Process Flows in any other Projects.

Process Flows

A Process Flow is used to organize one or more Streams into a logical collection of activities that has a start point, sequence of steps, and logical end point that describe a particular business process. Some examples of these business processes could be *Placing an Order, Billing Inquiries, Arranging Payments,* and so on.

Projects

A Project is used to combine Process Flows that relate to a common business function. For example, the *Sales Support Group* in a contact center may handle a number of different call types, each with its own unique Process Flow. So, *Sales Support Group* might be a Project containing the following Process Flows:

- Taking New Orders
- Tracking Orders
- Making Changes to Orders
- Handling Product Returns

Example: A Cable Service Contact Center Business Function

Let's look at an example of a contact center that handles calls for a cable service.

Assume that the cable service provides services for both residential and business customers including, basic cable television, digital cable television, cable internet, and so on.

The contact center for this cable service handles several customer call types:

- New Service
- Change or Upgrade Service
- Billing Inquiries
- Troubleshooting and Repair
- Cancellation of Service

The Project Books

We can create one Project Book for this cable service to handle the Residential Cable Business, and perhaps a second Project Book to handle the Commercial Cable Business.

The Projects

Next, within each Project Book we can create one Project for Cable Television Service, and a second Project for Cable Internet Service.

The Process Flows

Then, within each Project we can create Process Flows for each of the customer call types (New Service, Change or Upgrade Service, and so on).

The Streams

Within each Process Flow, we can identify individual Streams, or subprocesses, that combine together to handle the appropriate customer call type from start to finish. For example, in the New Service Process Flow, we might create four Streams to handle a New Service interaction:

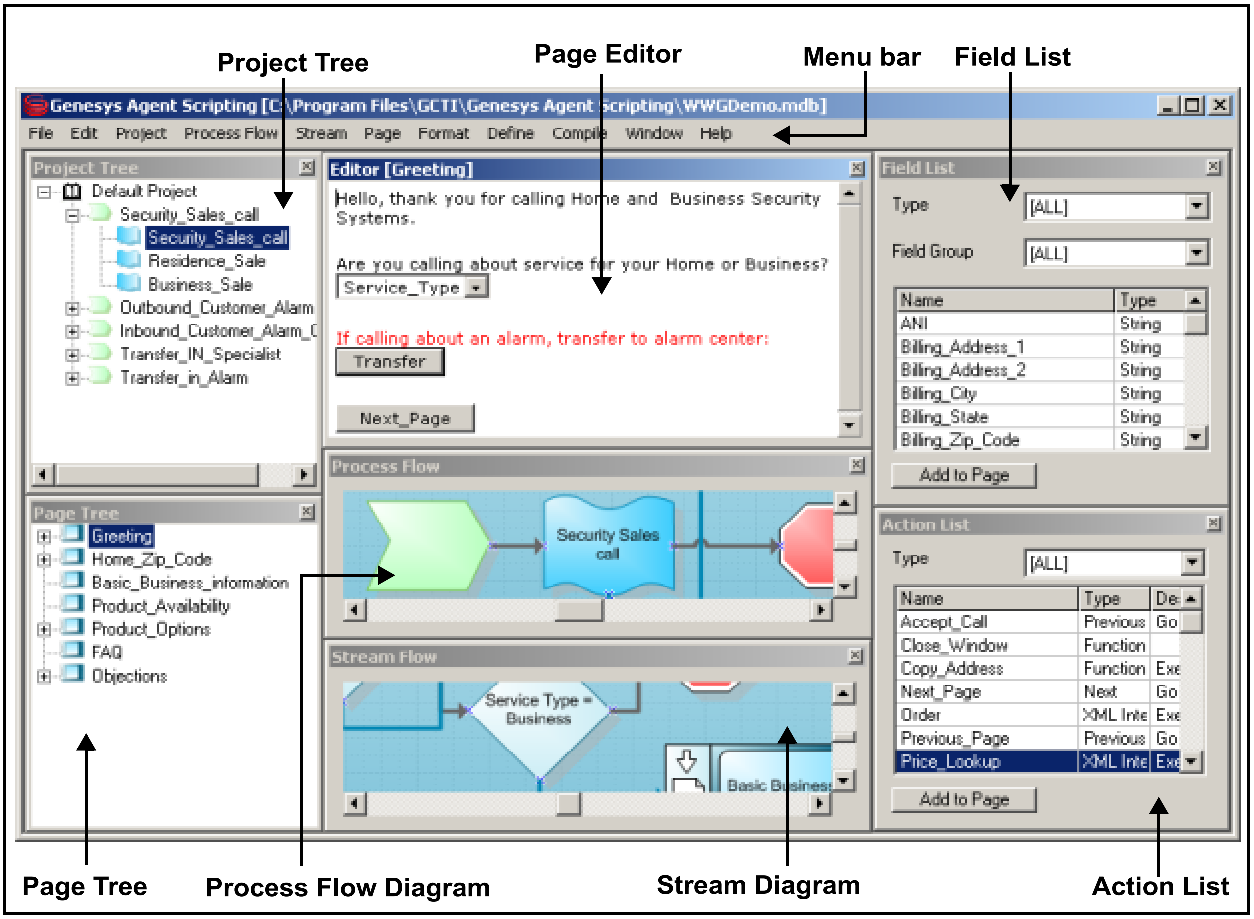
- Qualify the Customer (to determine if services are available at the customer's location)
- Select the Service
- Schedule Installation
- Arrange Payment

The Pages

Finally, within each Stream we create the individual web pages that make up each subprocess, and apply Branching Logic to implement the desired navigational structure. For example, the Qualify the Customer Stream might consist of Pages that collect customer information such as name, address, zip code, telephone number, and so on, and also Pages that request a database search to see which cable services, if any, are offered in the customer's region.

Desktop Layout of the Development Environment

Figure 1shows an example of the Genesys Agent Scripting Development Environment main window.



Hello, thank you for calling Home and Business Security Systems.

Are you calling about service for your Home or Business?

If calling about an alarm, transfer to alarm center:

Name	Type
ANI	String
Billing_Address_1	String
Billing_Address_2	String
Billing_City	String
Billing_State	String
Billing_Zip_Code	String

Name	Type	De
Accept_Call	Previous	Go
Close_Window	Function	
Copy_Address	Function	Exe
Next_Page	Next	Go
Order	XML Inte	Exe
Previous_Page	Previous	Go
Price_Lookup	XML Inte	Exe

Figure 1: Genesys Agent Scripting Development Environment

The desktop layout of the Agent Scripting Development Environment consists of:

- A set of menus (Menu bar)
- Project Tree
- Page Tree
- Page Editor
- Field List
- Action List
- Process Flow Diagram
- Stream Diagram

Menus on the Menu Bar

Genesys Agent Scripting provides the following menus:

- File
- Edit
- Project
- Process Flow
- Stream
- Page (and Page Layout)
- Format
- Define
- Compile
- Window
- Help

Many of the options available from menus are also available through shortcut menus when you right-click an item in the `Project Tree` or `Page Tree`.

Note: The tables in the subsections below introduce those menu options that specifically apply to the content of this User's Guide. Please see the *Genesys Agent Scripting Help* for more details on all menu options.

File Menu

Table 1 shows the options in the `File` menu that are discussed or referred to in this document.

Table 1: File Menu - Key Options

Menu Option	Description / What to Do...
Settings	Select File > Settings to open the Settings dialog box where you configure the overall system settings. See the section "Reviewing Project Book Settings" on page 60 for more details on the Settings dialog box.
Convert to Collaborate Mode *or* Collaborate	Select File > Convert to Collaborate Mode or File > Collaborate. Collaborate mode allows you to convert databases to multi-user format, check in and check out files, lock and unlock databases, and access the Administrate function to manage users and servers (for system administrators only). If the database you are working with has not been converted to collaborate mode, the File menu option reads Convert to Collaborate Mode. If the database has been converted to collaborate mode, the File menu option reads Collaborate, and contains a submenu with several additional options. Refer to the *Genesys Agent Scripting 7 Deployment Guide* for instructions on collaboration.
Open Project Book	Select File > Open Project Book for the Open Project Book dialog box from which you can select an existing Genesys Agent Scripting Project Book.Figure 3 on page 52 shows the Open Project Book dialog box.
New Project Book	Select File > New Project Book to open the New Genesys Agent Scripting Project Book dialog box from which you create your new Project Book. Project Book data resides in a file of type Microsoft Access (.mdb). For example, a Project Book called MyNewProject is stored in MyNewProject.mdb.
Import...	Select File > Import to open the Import From Project Book dialog box which enables you to import objects from other Genesys Agent Scripting Project Books into the Project Book currently in use. See the section "Importing Objects from Other Project Books" for more details.

Table 1: File Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Page Layouts >	The Page Layouts menu enables you to work with existing Page Layouts or create a new one. See the section “Creating a Page Layout” on page 303 and “Using the Page Layout Editor” on page 305 for more details.
Insert New Page Layout	Select File > Page Layouts > Insert New Page Layout to open the Page Layout dialog box where you define the characteristics of a new Page Layout. To open the Page Layout Editor to display and modify the new Page Layout, you select the Page Layout by name from File > Page Layouts.
Default_PageLayout	Select File > Page Layouts > Default_PageLayout to open the default Page Layout that was created when you created the Project Book. You can view and modify the default Page Layout as desired. Default_PageLayout may have a different name if you renamed the default Page Layout at any time.
<named Page Layouts that were created>	To display an existing Page Layout, select the Page Layout to be viewed from the list. The Page Layout will be displayed in the Genesys Agent Scripting Page Layout Editor.
User Defined Functions	Select File > User Defined Functions to open the User Defined Function List dialog box and add custom functions to a file called WWGUDFunctions.xxx. Then, you must add the function to Genesys Agent Scripting. Refer to “Creating User Defined Functions” on page 311 for additional detail.

Edit Menu

The Edit menu contains standard Windows editing commands and an option to reload the current Page, which clears all changes made to the Page since the last time the Page was loaded.

Project Menu

Table 2shows the options in the Project menu that are discussed or referred to in this document.

Table 2: Project Menu - Key Options

Menu Option	Description / What to Do...
Define Project	Select `Project` > `Define Project` to open the `Project List` dialog box, which enables you to create a new Project, edit existing Projects, create copies of Projects, and delete Projects.
Edit Project	Select `Project` > `Edit Project` to open the `Project Properties` dialog box, which enables you to edit an existing Project. (This dialog box also opens if you select `Project` > `Define Project`, select a Project, and click `Edit.`) If a Process Flow or Stream is selected in the `Project Tree` and you select the `Edit Project` menu option, the Project under which the Process Flow or Stream falls will be the one edited.
Save As...	Select `Project` > `Save As` to save an existing Project under a different name or with different details without overwriting the original.
Insert Process Flow	Select `Project` > `Insert Process Flow` to open the `Process Flow List` dialog box from which you add Process Flows to the selected Project.

Process Flow Menu

Table 3shows the options in the Process Flow menu that are discussed or referred to in this document.

Table 3: Process Flow Menu - Key Options

Menu Option	Description / What to Do...
Insert Process Flow	Select Process Flow > Insert Process Flow to open the Process Flow List dialog box, which enables you to create a new Process Flow, edit existing Process Flows, create copies of Process Flows, and delete Process Flows.
Edit Process Flow	Select Process Flow > Edit Process Flow to open the Process Flow Properties dialog box, which enables you to edit an existing Process Flow. (This dialog box also opens if you select Process Flow > Insert Process Flow, select a Process Flow, and click Edit.) If a Stream is selected in the Project Tree and you select the Edit Process Flow menu option, the Process Flow under which the Stream falls will be the one edited.
Save As...	Select Process Flow > Save As to save an existing Process Flow under a different name or with different details without overwriting the original.
Remove Process Flow	Select Process Flow > Remove Process Flow to remove the selected instance of the Process Flow from the Project Tree.
Insert Stream	Select Process Flow > Insert Stream to open the Stream List dialog box from which you add Streams to the selected Process Flow.

Stream Menu

Table 4shows the options in the Stream menu that are discussed or referred to in this document.

Table 4: Stream Menu - Key Options

Menu Option	Description / What to Do...
Insert Stream	Select Stream > Insert Stream to open the Stream List dialog box, which enables you to create a new Stream, edit existing Streams, create copies of Streams, and delete Streams.
Edit Stream	Select Stream > Edit Stream to open the Stream Properties dialog box, which enables you to edit an existing Stream. (This dialog box also opens if you select Stream > Insert Stream, select a Stream, and click Edit.) If a Page is selected in the Page Tree and you select the Edit Stream menu option, the Stream under which the Page falls will be the one edited.
Save As...	Select Stream > Save As to save an existing Stream under a different name or with different details without overwriting the original.
Remove Stream	Select Stream > Remove Stream to remove the selected instance of the Stream from the Project Tree.
Insert Page	Select Stream > Insert Page to open the Page List dialog box from which you add Pages to the selected Stream.

Page / Page Layout Menu

When the Page Editor is active, the Page menu is available. When the Page Layout Editor is active, the Page Layout menu replaces the Page menu.

Page Menu

Table 5shows the options in the Page menu that are discussed or referred to in this document.

Table 5: Page Menu - Key Options

Menu Option	Description / What to Do...
Insert Page	Select Page > Insert Page to open the Page List dialog box, which enables you to add a Page to a Stream as well as create new Pages, edit existing Pages, create copies of existing Pages, and delete Pages. The Page List dialog box lists Pages that have been defined but are not currently part of the selected Stream. To add a Page to the Stream, select the Page in the list and click OK. Inserting Pages from the Page menu will insert the Page above the currently selected Page in a Stream.
Edit Page	Select Page > Edit Page to open the Page Properties dialog box, which enables you to edit an existing Page. (This dialog box also opens if you select Page > Insert Page, select a Page, and click Edit.)
Save As...	Select Page > Save As to save an existing Page under a different name or with different details without overwriting the original.
Remove Page	Select Page > Remove Page to remove the selected instance of the Page from the Page Tree. This option does not delete the Page; it just removes it from the selected Stream.
Insert Branch	Select Page > Insert Branch to open the Branch Condition dialog box, which enables you to add branching conditions to the selected Page. This dialog box is also available by right-clicking a branching condition in the Page Tree. You may also *edit* a branch and *delete* a branch by right-clicking a branching condition in the Page Tree.

Table 5: Page Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Insert Field	Select Page > Insert Field to open the Field List dialog box, which enables you to place data Fields on the selected Page at the location of the cursor.
Insert Action	Select Page > Insert Action to open the Action List dialog box, which enables you to place Action buttons on the selected Page at the location of the cursor.
Insert Page Link	Select Page > Insert Page Link to open the Page Link Properties dialog box, which enables you to to add a link to other Genesys Agent Scripting Pages on the selected Page.
Insert HTML Table	Select Page > Insert HTML Table to open the Table dialog box, which enables you to add an HTML table structure to the selected Page. Once a table has been inserted, you can double-click the table in the script Page and make additional changes.
Insert URL Link	Select Page > Insert URL Link to add a link to other pages. Selecting this option opens the Link Properties dialog box.
Insert Navigator	Select Page > Insert Navigator to display a drop-down list of the set of Process Flows to which the script user can branch.
Insert Image	Select Page > Insert Image to open the Image dialog box, which enables you to place an image on the Page. Once selected, images are automatically copied to the images directory indicated in the Settings dialog box.
Insert Break	Select Page > Insert Break to move text and objects to a new line without starting a new paragraph. By default, Genesys Agent Scripting places a new paragraph mark in a script page when you select the Enter key.

Table 5: Page Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Edit HTML *or*	Select Page > Edit HTML to access and modify the HTML code that makes up the Page. When you select this option, the Genesys Agent Scripting Page Editor displays the HTML code rather than the Page content.
Edit Page View	Select Page > Edit Page View to display the Page content in the Genesys Agent Scripting Page Editor rather than the HTML code. This option only appears in the Page menu after Edit HTML has been selected. It takes the place of the Edit HTML menu option.

Page Layout Menu

Table 6shows the options in the Page Layout menu that are discussed or referred to in this document.

Table 6: Page Layout Menu - Key Options

Menu Option	Description / What to Do...
Delete Page Layout	Select Page Layout > Delete Page Layout to delete the Page Layout currently displayed in the Page Layout Editor. You must confirm your request to delete. If any objects currently use this Page Layout, Agent Scripting does not allow the deletion. You can see the objects in a cross reference list, and you must remove references between the objects and this Page Layout before you can delete the Page Layout.
Edit Page Layout Properties	Select Page Layout > Edit Page Layout Properties to open the Page Layout dialog box, which enables you to modify the selected Page Layout.
Save As	Select Page Layout > Save As to save an existing Page Layout under a different name without overwriting the original. The Page Layout dialog box appears with a Save As button instead of OK. Adjust the properties as necessary, then click Save As to save a copy of the Page Layout.

Table 6: Page Layout Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Insert Page Link	Select Page Layout > Insert Page Link to open the Page Link Properties dialog box, which enables you to add a link to other Genesys Agent Scripting Pages on the selected Page Layout at the location of the cursor.
Insert Page Area	The Page Area is where the contents of the created Pages appear in the web browser. Select Page Layout > Insert Page Area to add a Page Area to the Page Layout at the location of the cursor.
Insert Catalog Area	The Catalog Area is where the Page navigation information appears in the web browser. Select Page Layout > Insert Catalog Area to add a Catalog Area to the Page Layout at the location of the cursor.
Insert Image	Select Page Layout > Insert Image to open the Image dialog box, which enables you to add an image to the Page Layout at the location of the cursor. Once selected, images are automatically copied to the images directory indicated in the Settings dialog box.
Set Background Image	Select Page Layout > Set Background Image to open the Image dialog box, which enables you to add a *background* image to the Page Layout. Once selected, images are automatically copied to the images directory indicated in the Settings dialog box.

Table 6: Page Layout Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Edit HTML *or* Edit Page Layout View	Select Page Layout > Edit HTML to access and modify the HTML code that makes up the Page Layout. When you select this option, the Genesys Agent Scripting Page Layout Editor displays the HTML code rather than the Page Layout content. **Note:** Edit HTML is for advanced, experienced designers only. Use this feature carefully because you may overwrite generated code.
	Select Page Layout > Edit Page Layout View to display the Page Layout content in the Genesys Agent Scripting Page Layout Editor rather than the HTML code. This option only appears in the Page Layout menu after Edit HTML has been selected. It takes the place of the Edit HTML menu option.

Format Menu

Table 7shows the options in the Format menu that are discussed or referred to in this document. The Format menu is used to format text on the selected script Page.

Table 7: Format Menu - Key Options

Menu Option	Description / What to Do...
Bold	This option will set the font style of the selected text to **bold**.
Italic	This option will set the font style of the selected text to *italic*.
Underline	This option will set the font style of the selected text to underline.
Set Color	Select Format > Set Color to open the Color dialog box. This dialog box is the basic Microsoft Windows color selection dialog box. Select any color from the basic colors or define custom colors. When you click OK, the selected text in the script will be changed to the chosen color.

Table 7: Format Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Set Font	Select Format > Set Font to open the Select Font dialog box. This dialog box allows you to select a font for text items on the Page. Select font options and click OK to change the selected text.
Justify >	The Justify option allows you to align text on the Page.
Left	Select Format > Justify > Left to align the selected text with the left margin.
Center	Select Format > Justify > Center to center the selected text on the Page.
Right	Select Format > Justify > Right to align the selected text with the right margin.
Justify	Select Format > Justify > Justify to space the words on a line equally so that the line appears to be aligned with both the left and right margins. The last line of the paragraph will be left justified.
HTML Table >	Select Format > HTML Table for options that enable you to modify an HTML table you've selected on the Page. See the "Formatting HTML Tables" on page 162 section for more details.
Add Row	Select Format > HTML Table > Add Row to add a new row above the selected row(s).
Delete Rows	Select Format > HTML Table > Delete Rows to delete the selected row(s).
Add Column	Select Format > HTML Table > Add Column to add a column to the left of the selected column.
Merge Columns	Select Format > HTML Table > Merge Columns to merge the selected cells into one cell.
Split Columns	Select Format > HTML Table > Split Columns to divide the selected cell(s).

Table 7: Format Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Delete Columns	Select Format > HTML Table > Delete Columns to delete the selected column(s).
Add Cells	Select Format > HTML Table > Add Cell to add a cell to the left of the selected cell.
Delete Cells	Select Format > HTML Table > Delete Cell to delete the selected cell(s).
Toggle Border	Select Format > HTML Table > Toggle Border to add or remove the border from the selected table.
List	Select Format > List to convert selected text to an ordered list.

Define Menu

Table 8shows the options in the Define menu that are discussed or referred to in this document.

Table 8: Define Menu - Key Options

Menu Option	Description / What to Do...
Projects	Select Define > Projects to open the Project List dialog box, which enables you to create a new Project, edit existing Projects, create copies of Projects, and delete Projects.
Process Flows	Select Define > Process Flows to open the Process Flow List dialog box, which enables you to create a new Process Flow, edit existing Process Flows, create copies of Process Flows, and delete Process Flows.
Streams	Select Define > Streams to open the Stream List dialog box, which enables you to create a new Stream, edit existing Streams, create copies of Streams, and delete Streams.
Pages	Select Define > Pages to open the Page List dialog box, which enables you to create new Pages, edit existing Pages, create copies of existing Pages, and delete Pages. See the chapter called "Using the Page Editor" for more details.

Table 8: Define Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Fields	Select `Define` > `Fields` to open the `Field List` dialog box, which enables you to create new Fields, edit existing Fields, create copies of existing Fields, and delete Fields. See the chapter called "Creating Fields" for more details.
Actions	Select `Define` > `Actions` to open the `Action List` dialog box, which enables you to create new Actions, edit existing Actions, create copies of existing Actions, and delete Actions. See the chapter called "Creating Actions" for more details.
XML Interfaces	Select `Define` > `XML Interfaces` to open the `XML Interface List` dialog box, which enables you to create new XML Interfaces, edit existing XML Interfaces, create copies of existing XML Interfaces, and delete XML Interfaces.
Databases	Select `Define` > `Databases` to open the `Database List` dialog box, which enables you to create new Database connections, edit existing Database connections, create copies of existing Database connections, and delete Database connections. See the chapter called "Databases and Database Interfaces" for more details.
Database Interfaces	Select `Define` > `Database Interfaces` to open the `Database Interface List` dialog box, which enables you to create new Database Interfaces, edit existing Database Interfaces, create copies of existing Database Interfaces, and delete Database Interfaces. See the chapter called "Databases and Database Interfaces" for more details.
API Interfaces	Select `Define` > `API Interfaces` to open the `API Interface List` dialog box, which enables you to create new API Interfaces, edit existing API Interfaces, create copies of existing API Interfaces, and delete API Interfaces.

Table 8: Define Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Async Interfaces	Select `Define` > `Async Interfaces` to open the `Async Interface List` dialog box, which enables you to create new Async Interfaces, edit existing Async Interfaces, create copies of existing Async Interfaces, and delete Async Interfaces.
Catalogs	Select `Define` > `Catalogs` to open the `Catalog List` dialog box, which enables you to create new Catalogs, edit existing Catalogs, create copies of existing Catalogs, and delete Catalogs. See the section “Creating a Catalog” on page 295 for more details.
Templates	Select `Define` > `Templates` to open the `Template List` dialog box, which enables you to create new Templates, edit existing Templates, create copies of existing Templates, and delete Templates.
Custom Field Types	Select `Define` > `Custom Field Types` to open the `Custom-Field-Type List` dialog box, which enables you to create new Custom Field types, edit existing Custom Field types, create copies of existing Custom Field types, and delete Custom Field types.

Compile Menu

Table 9shows the options in the Compile menu that are discussed or referred to in this document.

Table 9: Compile Menu - Key Options

Menu Option	Description / What to Do...
Target Environment	Select Compile > Target Environment to open the Target Environment List dialog box, which enables you to create new Target Environments, edit existing Target Environments, create copies of existing Target Environments, and delete Target Environments. See the section "Creating a Target Environment" on page 270 for more details.
Compile	Select Compile > Compile to open the Compile dialog box, which enables Genesys Agent Scripting to produce ASP, ASPX, or JSP code from the script Pages you created. You can also run Simulate from this dialog box as well. See the section "Compiling a Script" on page 281 more details.
Simulate	Select Compile > Simulate after you have finished editing the scripts and have compiled them. This immediately opens a web browser and begins the simulation. See the section "Testing the Compiled Code with Simulate" on page 283 in for more details.

Table 9: Compile Menu - Key Options (Continued)

Menu Option	Description / What to Do...
Trace Options	Select Compile > Trace Options to open the Trace Options dialog box, which allows you to customize your logging options such as level, size, and number of trace files. For more information please consult the *Genesys Agent Scripting Help.*
Assignment Rules	This option is only available for Project Books with Genesys Integration. Select Compile > Assignment Rules to open the Process Flow Assignment Rules dialog box, which allows you to define assignment rule expressions for a given Process Flow. Assignment Rules are used in conjunction with scripts that are integrated with the Genesys Agent Interaction Layer. Data associated with the interaction may be used by the Assigner Servlet (Genesys Agent Desktop deployment) or by Interaction Workspace Plug-in for Agent Scripting (Interaction Workspace deployment), based on Assignment Rules to determine the correct Process Flow to launch for the interaction. • Genesys Agent Desktop deploys Assignment Rules as a servlet running on the Genesys Agent Desktop site. • Interaction Workspace stores Assignment Rules as a Script of the corresponding tenant.

Window Menu

The Window menu is used to manage the different windows within Genesys Agent Scripting. Table 10shows the key options in this menu.

Table 10: Window Menu - Key Options

Menu Option	Description / What to Do...
Default Setup	Windows within Genesys Agent Scripting can be moved, resized, and closed. Select Window > Default Setup to return the windows to their default positions and sizes relative to the main Genesys Agent Scripting window.
Open Tools >	The Open Tools menu option allows you to reopen any window that has been closed. This option shows a list of all of the main Genesys Agent Scripting windows.
Project Tree Page Tree Process Flow Stream Flow Editor Field List Action List	A check mark beside a window name indicates that the window is currently open. Selecting an unchecked window will open that window.
[list of open windows]	Below Open Tools is a list of all Genesys Agent Scripting windows that are currently open. A check mark is displayed beside the currently active window. You can select a different window in this list to change which window is active.

Help Menu

Table 11shows the options in the Help menu that are discussed or referred to in this document.

Table 11: Help Menu - Key Options

Menu Option	Description / What to Do...
Genesys Agent Scripting Help	Select Help > Genesys Agent Scripting Help to open a Help window which contains reference information about Genesys Agent Scripting and the Development Environment.
About Genesys Agent Scripting...	Select Help > About Genesys Agent Scripting to open the About Genesys Agent Scripting dialog box. This contains product version and licensing information, as well as buttons to view system information and to register the product. Product version and system information are useful if you need to call for technical support.

Project Tree

The Project Tree provides an outline (tree view) of the Projects, Process Flows, and Streams in a Project Book.

Icons appear next to the text at each level in the tree indicating what type of object is at each level.

Page Tree

When you select a Stream in the Project Tree, you will see the Pages associated with that Stream in the Page Tree.

The Page Tree contains a tree view of Pages and their branching conditions that are contained within the Stream selected in the Project Tree.

Icons appear next to the text at each level in the tree indicating what type of object is at each level.

Process Flow

When you select Window > Open Tools > Process Flow, the Process Flow window becomes the currently active window and causes a check to appear next to it. A Process Flow diagram shows the collection of Streams that comprise that Process Flow, a default flow, any branches connecting the Streams, and any branches connecting to other Process Flows.

Right-clicking within a Process Flow window opens a submenu where you can select:

- Layout Shapes
- Print Diagram
- Insert Stream

Stream

When you select Window > Open Tools > Stream, the Stream window becomes the currently active window and causes a check to appear next to it. A Stream diagram shows the flow from Page to Page within a Stream including all branching conditions within the Stream as well as connections to other Streams and Process Flows.

In addition, any Actions that occur during Page Load or Page Unload are indicated by a star icon.

Right-clicking within a Stream window opens a submenu where you can select:

- Layout Shapes
- Print Diagram
- Insert Page

Editor

The Page Editor shows the content of individual web pages in a Stream. The Page Editor for a particular Page is accessed by clicking the desired Page in the Page Tree.

In the Page Editor, you can change information, add text, delete text, cut and paste text or images, add Fields and Actions from the Field List (see “Field List”) and Action List (see “Action List”),. and edit HTML code if you are comfortable with HTML. See the chapter called “Using the Page Editor” on page 151 for more details about the Page Editor.

Note: Under certain conditions, the area occupied by the Page Editor becomes the Page Layout Editor. See the section “Creating a Page Layout” on page 303 and “Using the Page Layout Editor” on page 305 for more details about the Page Layout Editor.

Field List

The Field List shows all the Fields defined in the open Project Book.

You can filter the Fields that appear in the list using the Type dropdown list within the Field List.

You can add a Field to the Page you are currently editing by selecting a Field in the Field List and clicking the Add to Page button at the bottom of the Field List.

You can also group Fields by a Field Group name. You can choose to display only the Fields belonging to a particular group (such as Customer Field group).

Action List

The Action List shows all the Actions defined in the open Project Book.

You can filter the Actions that appear in the list using the Type dropdown list within the Action List.

You can add an Action to the Page you are currently editing by selecting an Action in the Action List and clicking the Add to Page button at the bottom of the Action List.

Putting it All Together

Genesys Agent Scripting offers a quick and easy way to get started creating the hierarchical structure you need for an application. As you will see in “Creating a New Project Book” on page 51, the Set Up a New Project wizard helps you create a default Project, default Process Flow, default Stream, default Page, and other necessary organizational elements to get you started.

You then create additional Projects, Process Flows, Streams, and Pages as desired using the Project List, Process Flow List, Stream List, and Page List dialog boxes as appropriate.

You then need to insert each element you create into the next-highest element to which it belongs. For example:

- Process Flows can be inserted into the appropriate Project using the Project menu (see “Project Menu” on page 25).
- Streams can be inserted into the appropriate Process Flow using the Process Flow menu (see “Process Flow Menu” on page 27).
- Individual Pages can be inserted into the appropriate Stream using the Stream menu (see “Stream Menu” on page 28).

Likewise, when adding information or objects to a Page, such as branches, Fields, Actions (which are all described in their own chapters), you will use the Page menu for this purpose (see “Page Menu” on page 29).

The Project Tree shows you the hierarchical structures you have created. You are free to reorder elements as desired.

The Page Tree shows you the Pages that belong to the Stream that is selected in the Project Tree. You are free to reorder these Pages as desired as well.

See the section "Desktop Layout of the Development Environment" on page 22 for more details on the Genesys Agent Scripting Development Environment user interface.

Chapter

2 Application Flow

This chapter provides a brief overview of a typical high-level process a script developer might follow to use the Genesys Agent Scripting Development Environment to build scripts, and compile and test applications.

The information in this chapter is divided among the following topics:

The Process of Building Scripts

Genesys Agent Scripting is a powerful and extensive tool for building scripts and script applications. The user interface of the Genesys Agent Scripting Development Environment contains numerous menus, windows, and dialog boxes, and the software itself has many basic and advanced features and options that help open the task of script development and deployment to a larger number of individuals than might normally undertake such a task.

Every individual script developer will have his or her own preferred way of using Genesys Agent Scripting to develop applications. Likewise, the software is flexible enough to allow many different approaches to be used successfully. However, the purpose of this brief chapter is to provide an example of how a typical application development flow might proceed.

The process of building scripts involves working through a series of tasks in an organized, progressive fashion. Figure 2 on page 50shows a diagram which captures the most significant steps listed below.

Creating a Project Book

You normally begin by creating a new Project Book. You can use the `Set Up a New Project` wizard to automate this process to a great degree.

See "Creating a New Project Book".

Creating Projects, Process Flows, and Streams

Your new Project Book will contain a default Project, default Process Flow, and default Stream. However, if you have followed a process similar to that described in "Organizing a New Project" on page 19 of the chapter called "Overview of the Development Environment," you may want to define additional Projects, Process Flows, or Streams at this time.

These tasks can also occur at a later time if you so choose.

Each Process Flow has a default Page order associated with it. The first Page of the first Stream, as seen in the `Project Tree` and `Page Tree`, is the default starting Page. In the absence of any other overriding branching condition, the next logical page will be the next Page as defined in the `Page Tree`, until the end of the Stream is reached. When the end of the Stream is reached, the next logical page will be the first Page of the next Stream, as defined in the `Project Tree`. This Page flow continues until the last Page of the last Stream for a given Process Flow is reached.

Defining Fields

Before creating and working with the individual Pages that make up a Stream, it is usually a good idea to define some or all of the Fields that will eventually be added to these Pages. If you have designed and planned your application ahead of time, it should be possible to do this. Then, you can simply add the Fields to the appropriate Pages as you create those Pages.

See "Creating Fields" beginning on page 65.

Defining Actions

Before creating and working with the individual Pages that make up a Stream, it is usually a good idea to define some or all of the Actions that will eventually be added to these Pages. If you have designed and planned your application ahead of time, it should be possible to do this. Then, you can simply add the Actions to the appropriate Pages as you create those Pages.

See "Creating Actions" beginning on page 109.

Defining Database Connections

Some Actions require reading from or writing to an external database. While you are defining Actions, it is reasonable to define the Database connections that Genesys Agent Scripting requires to communicate with the specific external databases you need in your script application.

See “Creating a Database Connection” on page 173 of the chapter called “Databases and Database Interfaces.”

Defining Database Interfaces

Once a Database connection has been set up, you need to define specific Database Interfaces that dictate how Genesys Agent Scripting should operate on the external database. These are queries that select data from a database, insert data into a database, or update a database.

See “What Is a Database Interface?” on page 172 of the chapter called “Databases and Database Interfaces.”

Creating Pages

Creating and developing the individual script Pages will probably occupy most of your development time when using Genesys Agent Scripting. Here is where you design each Page and add the text, images, Fields, and Actions that bring the Page to life during execution.

See “Using the Page Editor” beginning on page 151.

Creating Branching Statements

Branching statements connect Pages together to create sequences which script users work with. Branches are a set of logic that directs the flow from one Page to the next within the various Process Flows. They provide the map through an interaction that is followed when Action buttons are clicked.

See “Adding Branching Logic” beginning on page 257.

Creating and Editing a Page Layout

To obtain a unique or common look-and-feel for all Pages or for a sequence of Pages in your script application, you can create a new Page Layout or modify the existing default Page Layout. Typical Page Layouts consist of a `Page Area`, `Catalog Area`, `Header`, and `Footer`. Any of these areas can be resized, moved, or deleted to create the specific arrangement you desire. In addition, you can add images to a Page Layout or a background image which will be displayed on every Page that uses that Page Layout.

See “Creating a Page Layout” on page 303 and “Using the Page Layout Editor” on page 305 of the chapter called “Advanced Topics”

Creating a Catalog

As you are developing your script application, you may find that having an area that holds navigational information common to every Page or to a sequence of Pages would be helpful to script users. This area is referred to as a *Catalog,* and it resides on the Page Layout. The default Page Layout contains a Catalog Area which can be resized or moved if desired. Also, you can add a Catalog to any new Page Layouts you create.

See “Creating a Catalog” on page 295 of the chapter called “Advanced Topics”

Defining or Selecting a Target Environment

A default Target Environment was set up when you created your new Project Book. However, before you can compile and test the script application you are developing, you must define or select a Target Environment.

The Target Environment specifies the directory to which the code for your script will be compiled, the type of code to generate (ASP, ASPX, or JSP), and other necessary information about your application’s development environment.

See “Creating a Target Environment” on page 270.

Compiling and Testing the Application

Compiling the code and testing the application with the `Simulate` function are tasks that can occur at any point in the process of script development, provided that a Target Environment has been set up and there are Pages to compile and test.

Compiling and testing is an iterative process which some script developers may choose to carry out frequently during development. Others may choose to compile and test less frequently, only after significant script development has taken place.

The `Compile` function takes the pages that you created and produces the Genesys Agent Scripting ASP, ASPX, or JSP code. It compiles all objects (for example, Pages, Fields, Actions) in the current Project Book.

The `Simulate` function always opens an Internet Explorer web browser and simulates execution of the application. This allows testing of the application’s functionality and deployment to assure that it performs correctly when put into production.

See Chapter 12, “Compiling, Testing, and Deploying Your Application,” on page 269.

Repeating the Process

Successful use of the Genesys Agent Scripting Environment involves repeating the processes described in preceding sections in a fashion that leads to the development of a working script application. As stated earlier, there are many ways you can choose to use this software to accomplish this goal, and Genesys Agent Scripting is flexible enough to handle a variety of approaches.

Deploying the Application for Production

Once you have completed script application development and testing, your application will be ready for production deployment to the target population for whom it is intended. In many organizations, this function may be carried out by a system administrator or individual other than the script developer. Production deployment involves uploading the generated pages and all their associated files to a web server such that, given the proper URL, a target script user (agent) can open a web browser and execute the application.

See “Deploying the Application” on page 286.

A Typical Application Development Flow

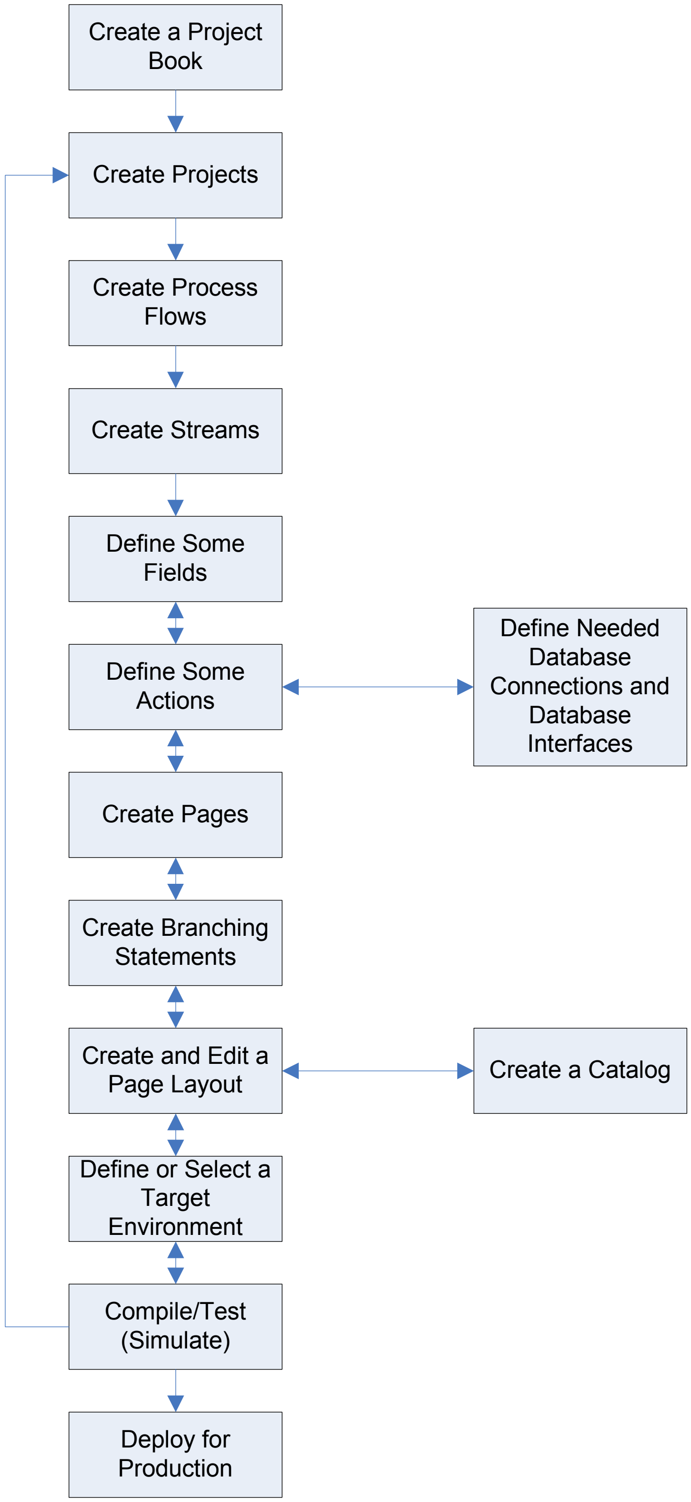


Figure 2: Typical Application Flow

Chapter

3 Creating a New Project Book

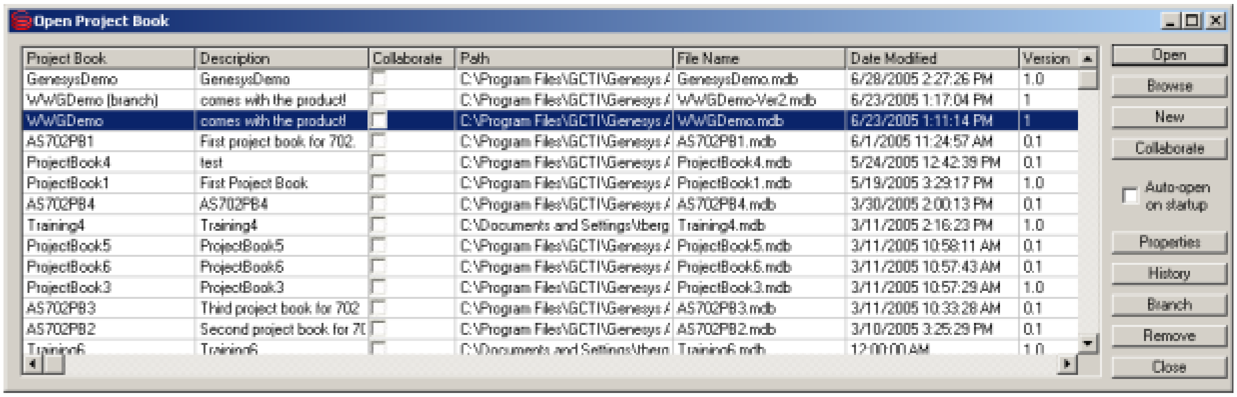
This chapter provides an overview of the process of creating a new Project Book and its components using the `Set Up a New Project` wizard.

The information in this chapter is divided among the following topics:

Starting the Genesys Agent Scripting Development Environment

The Genesys Agent Scripting Development Environment runs under Microsoft Windows. To start Agent Scripting from the Windows Start menu:

- Select `Start` > `Programs` > `Genesys Solutions` > `Genesys Agent Scripting` > `Start Genesys Agent Scripting`.
- The `Open Project Book` window opens as shown in Figure 3 on page 52.



Project Book	Description	Collaborate	Path	File Name	Date Modified	Version
GenesysDemo	GenesysDemo	□	C:\Program Files\GCTI\Genesys A	GenesysDemo.mdb	6/28/2005 2:27:26 PM	1.0
WWGDemo (branch)	comes with the product!	□	C:\Program Files\GCTI\Genesys A	WWGDemo-Ver2.mdb	6/23/2005 1:17:04 PM	1
WWGDemo	comes with the product!	□	C:\Program Files\GCTI\Genesys A	WWGDemo.mdb	6/23/2005 1:11:14 PM	1
AS702PB1	First project book for 702.	□	C:\Program Files\GCTI\Genesys A	AS702PB1.mdb	6/1/2005 11:24:57 AM	0.1
ProjectBook4	test	□	C:\Program Files\GCTI\Genesys A	ProjectBook4.mdb	5/24/2005 12:42:39 PM	0.1
ProjectBook1	First Project Book	□	C:\Program Files\GCTI\Genesys A	ProjectBook1.mdb	5/19/2005 3:29:17 PM	1.0
AS702PB4	AS702PB4	□	C:\Program Files\GCTI\Genesys A	AS702PB4.mdb	3/30/2005 2:00:13 PM	0.1
Training4	Training4	□	C:\Documents and Settings\tberg	Training4.mdb	3/11/2005 2:16:23 PM	1.0
ProjectBook5	ProjectBook5	□	C:\Program Files\GCTI\Genesys A	ProjectBook5.mdb	3/11/2005 10:58:11 AM	0.1
ProjectBook6	ProjectBook6	□	C:\Program Files\GCTI\Genesys A	ProjectBook6.mdb	3/11/2005 10:57:43 AM	0.1
ProjectBook3	ProjectBook3	□	C:\Program Files\GCTI\Genesys A	ProjectBook3.mdb	3/11/2005 10:57:29 AM	1.0
AS702PB3	Third project book for 702	□	C:\Program Files\GCTI\Genesys A	AS702PB3.mdb	3/11/2005 10:33:28 AM	0.1
AS702PB2	Second project book for 70	□	C:\Program Files\GCTI\Genesys A	AS702PB2.mdb	3/10/2005 3:25:29 PM	0.1
Training6	Training6	□	C:\Documents and Settings\tberg	Training6.mdb	12:00:00 AM	1.0

Figure 3: Open Project Book Window

You can create a new Project Book by clicking `New`. See the next sections for additional details.

If you have previously defined Project Books, you can open one and work with it by selecting the Project Book and clicking `Open`.

Creating the Project Book

Procedure: Create the Project Book

Purpose: A Project Book is a collection of Projects. Each Project is a collection of Process Flows, which usually represent a single interaction from start to finish. In general, you should use a Project Book to store related Projects or Process Flows.

There are two main methods of initiating the process of creating a Project Book:

- If you have just started Genesys Agent Scripting as described in the previous section, you can initiate the process from the `Open Project Book` window.
- If you are already in the Genesys Agent Scripting Development Environment, you can initiate the process from the `File` menu.

Start of procedure

1. Click `New` from the `Open Project Book` window (see Figure 3), or

   Select `File` > `New Project Book` from within the Genesys Agent Scripting Development Environment.

2. In the New Genesys Agent Scripting Project Book dialog box, type a name for this Project Book and click Save.

   This will be the name of the Microsoft Access database (the.mdb extension will be added automatically).

3. In the Project Book Properties dialog box, type a name in the Project Book field.

   This will be the Project Book name stored as part of the data used to manage projects and their versions. See the section “Managing Multiple Versions of Scripts” on page 340 for information on versioning.

4. You may select from the Base Toolkit dropdown list to initialize the new Project Book with one of the following Project Book Toolkits supplied by Agent Scripting:

   Agent Scripting Toolkit – Provides the new Project Book the functions to create scripts with reporting features and to suspend and resume scripts.

   Note: ASP scripts using the Agent Scripting Toolkit do not support Suspend and Resume actions and will return XML Interface warnings when compiled.

   Genesys Agent Interaction Toolkit – Provides the new Project Book the same functions as the Agent Scripting Toolkit along with Genesys Agent Interaction Layer integration functionality, such as Telephony Controls, Contact Data, Interaction Data, Agent Data, and Outbound Data.

   The Default is None, which indicates that the Project Book is not initialized with a toolkit.

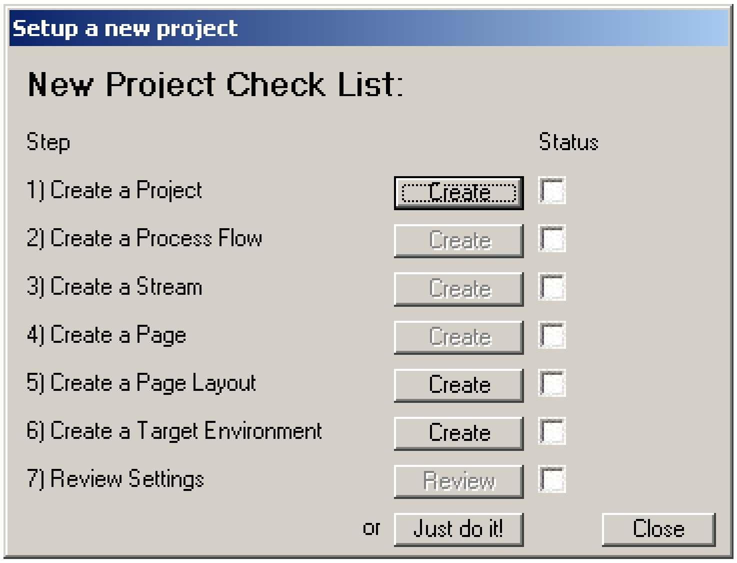
5. Type a brief description in the Description field, and a version number for this Project Book in the Version field, and then click OK.

End of procedure

Next Steps

Proceed to the next section to use the Set Up a New Project wizard.

Setting Up a New Project Wizard



Setup a new project

New Project Check List:

Step		Status
1) Create a Project	Create	□
2) Create a Process Flow	Create	□
3) Create a Stream	Create	□
4) Create a Page	Create	□
5) Create a Page Layout	Create	□
6) Create a Target Environment	Create	□
7) Review Settings	Review	□
or	Just do it!	Close

Figure 4: Set Up a New Project Wizard

There are several components of a Project Book that need to be set up. Genesys Agent Scripting facilitates this process through the Setup a New Project Wizard, or New Project Check List, as shown in Figure 4

Procedure: Set Up a New Project Wizard

Purpose: The fastest and easiest way to get started is to allow Genesys Agent Scripting to create all these components automatically. Several of the components (Project, Process Flow, Stream, Page, and Page Layout) will be created with their default values.

Start of procedure

1. Click Just Do It! to run the automatic process.
2. At the Target Environment Properties dialog box, you have the option to change any of the default values before proceeding. In our example, keep the defaults and click OK.

Note: If you selected integration with the `Genesys Agent Interaction Toolkit` for this Project Book, you can set the `Code Target` field to either `JSP` or `ASPX.` ASPX scripts created with the Genesys Agent Interaction Toolkit selected as the Base Toolkit, require Genesys Integration Server version 7.2 or higher in the Genesys environment.

The Genesys Agent Scripting Development Environment opens. The software has created a `Default_Project`, `Default_ProcessFlow`, `Initial_Stream`, and `Initial_Page` within the `Default_PageLayout` for you for the Project Book you have just defined. In addition, you have a default Target Environment to which scripts will be compiled.

If you selected integration with the Genesys suite for this Project Book, your `Field List` and `Action List` are populated with numerous Fields and Actions, respectively. These elements are designed for use with Genesys integration.

End of procedure

Your Project Book is defined. See the section "Reviewing Project Book Settings" on page 60 to review your default Project Book settings.

Creating a Project

Procedure: Create a Project

Purpose: As an alternative to the `Just Do It!` approach, you may create and define the components of your Project Book individually from the `Set Up a New Project` wizard shown in Figure 4 on page 54.

A Project is the highest level entity you will work with. It is a collection of Process Flows.

Start of procedure

1. Click the `Create` button beside `1) Create a Project` to open the `Project Properties` dialog box.
2. Type a Project name in the `Project` field, and optionally a description, then click `OK`.

The wizard (check list) `Status` check box beside `1) Create a Project` shows that this step is complete.

End of procedure

Creating a Process Flow

Procedure: Create a Process Flow

Purpose: A Process Flow is a logical collection of activities that has a starting point, a set of steps, and an ending point. For example, a Process Flow may include multiple activities that make up a business process, such as handling a service request, signing up a new customer, and so on.

Start of procedure

1. Click the `Create` button beside `2) Create a Process Flow` to open the `Process Flow Properties` dialog box.
2. Type a Process Flow label name in the `Label` field.
3. Click inside the `Name` field and the field is populated with a name corresponding to the label.
4. Optionally type a description, then click `OK`.
5. The wizard (check list) `Status` check box beside `2) Create a Process Flow` shows that this step is complete.

End of procedure

Creating a Stream

Procedure: Create a Stream

Purpose: A Stream is a sub-process that is likely to be reused, such as a credit card validation, customer lookup, and so on.

Start of procedure

1. Click the `Create` button beside `3) Create a Stream` to open the `Stream Properties` dialog box.
2. Type a Stream label name in the `Stream Label` field.

3. Click inside the `Stream Name` field and the field is populated with a name corresponding to the label.
4. Optionally type a description, then click `OK`.

   The wizard (check list) `Status` check box beside `3) Create a Stream` shows that this step is complete.

End of procedure

Creating a Page

Procedure:
Create a Page

Purpose: Pages represent the individual web pages that agents see when they run the compiled, completed script application through a web browser. Pages contain text, Fields, and Actions (Action buttons and/or Action code that is executed when a Page opens or closes).

Start of procedure

1. Click the `Create` button beside `4) Create a Page` to open the `Page Properties` dialog box.
2. Type a Page label name in the `Page Label` field.
3. Click inside the `Page Name` field and the field is populated with a name corresponding to the label.
4. Optionally type a description, then click `OK`.

Note: There are additional options on the `Page Properties` dialog box. At this time we will leave these options "as is."

Additional Page Properties are discussed in _in the table called "Page Properties" on page 154.

The wizard (check list) `Status` check box beside `4) Create a Page` shows that this step is complete.

End of procedure

Creating a Page Layout

Procedure: Create a Page Layout

Purpose: A Page Layout is a definition of the look and feel for the web pages that will be displayed.

Start of procedure

1. Click the `Create` button beside `5) Create a Page Layout` to open the `Page Layout` dialog box.
2. Type `Default` as the Page Layout name in the `Layout Name` field.
3. Optionally type a description, then click `OK`.

Note: There are additional options on the `Page Layout` dialog box. At this time we will leave these options "as is."

Additional Page Layout Properties are discussed in "Advanced Topics" in the section called "Creating a Page Layout" on page 303.

The wizard (check list) `Status` check box beside `5) Create a Page Layout` shows that this step is complete.

End of procedure

Creating a Target Environment

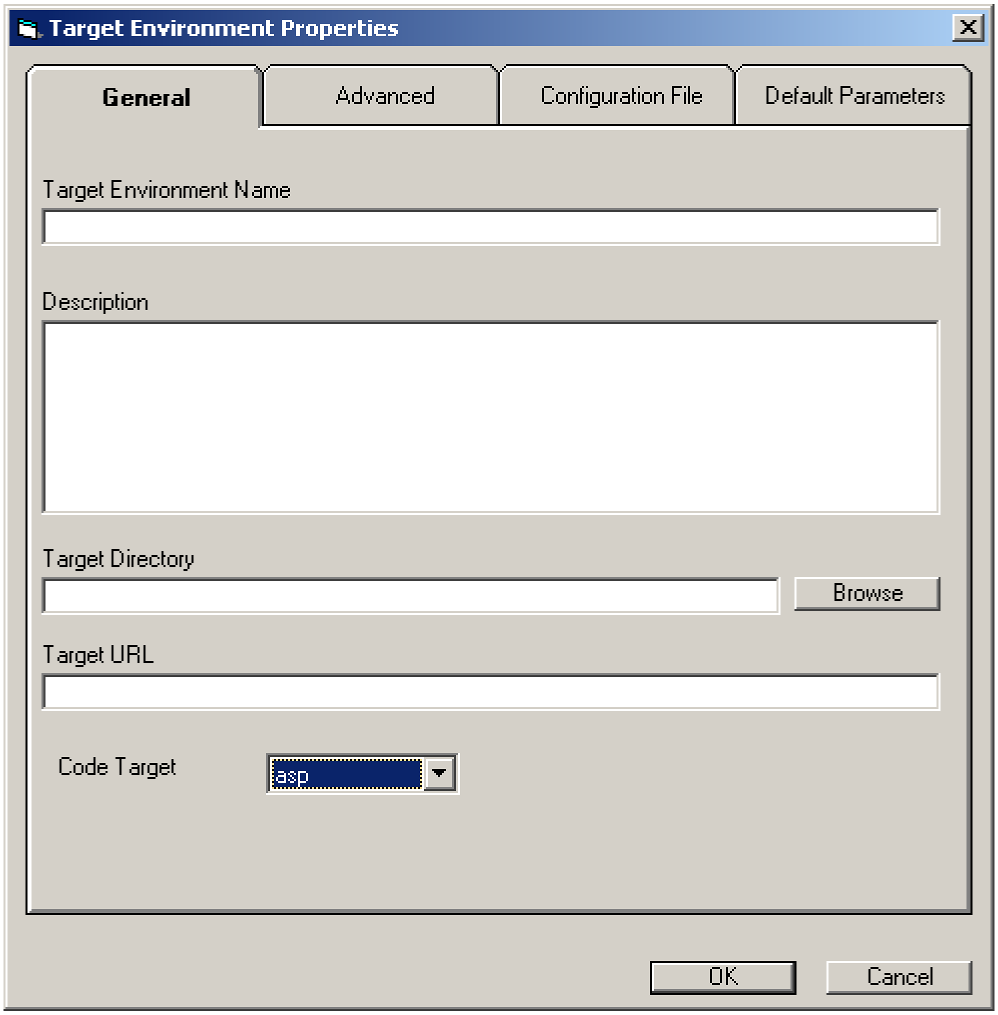
Procedure: Create a Target Environment

Purpose: The Target Environment is the directory to which the code will be compiled.

Start of procedure

1. Click the `Create` button beside `6) Create a Target Environment` to open the `Target Environment Properties` dialog box as shown in Figure 5.

   More information about the `Target Environment Properties` dialog box is provided in "Compiling, Testing, and Deploying Your Application."



Target Environment Name

Description

Target Directory

Target URL

Code Target asp

Figure 5: Target Environment Properties Dialog Box

2. Type a Target Environment name in the `Target Environment Name` field.
3. Optionally type a description.
4. Type a Target Directory path, or select one by clicking Browse and navigating to the appropriate directory.

   The Target Directory is where the Agent Scripting application is going to be deployed. Mapping a network drive is an easy way to deploy the application directly on a different machine running the web server. For example, if `w:` is mapped to a Microsoft IIS web server machine, then the Target Directory might look like:

   ```
   w:\Inetpub\wwwroot\new_app
   ```

   If deployed on a Tomcat web server, it might look like:

   ```
   w:\Program Files\Apache Software Foundation\Tomcat
   5.0\webapps\new_app
   ```

5. The `Target URL` is the address that will be in the browser when `Compile` > `Simulate` (or `Ctrl+F5`) is selected. This will start up the newly-deployed application.

   For an IIS deployment, you might have:

   `http://IIS_Server/new_app`

   For a Tomcat deployment, you might have:

   `http://Tomcat_Server:8080/new_app`

Note: When specifying a `Target URL` for your Target Environment, use the full computer name or IP address (http://servername or http://xxx.xxx.xxx.xxx). Do *not* use http://localhost.

6. If you know that you will be compiling Active Server Pages (ASPs), select the `asp` default in the `Code Target` drop-down list. ASP is used with Microsoft IIS Web servers.

   If you know that you will be compiling Java Server Pages (JSPs), select `jsp` in the `Code Target` drop-down list. JSP is used with Apache Tomcat Web servers and requires the presence of a Java Runtime Environment (JRE).

   Likewise, if you know that you will be compiling Active Server Pages - Extended (ASPXs), select `aspx` in the `Code Target` dropdown list. ASPX is used with Microsoft IIS and an installed Microsoft .NET Framework v1.1.

7. Click `OK`.

   The wizard (check list) `Status` check box beside `6) Create a Target Environment` shows that this step is complete.

End of procedure

Reviewing Project Book Settings

Procedure:
Review Project Book Settings

Purpose: The `Settings` dialog box defines default parameters for a Genesys Agent Scripting Project Book.

Start of procedure

1. Click the `Review` button beside `7) Review Settings` to open the `Settings` dialog box that is shown in Figure 6.

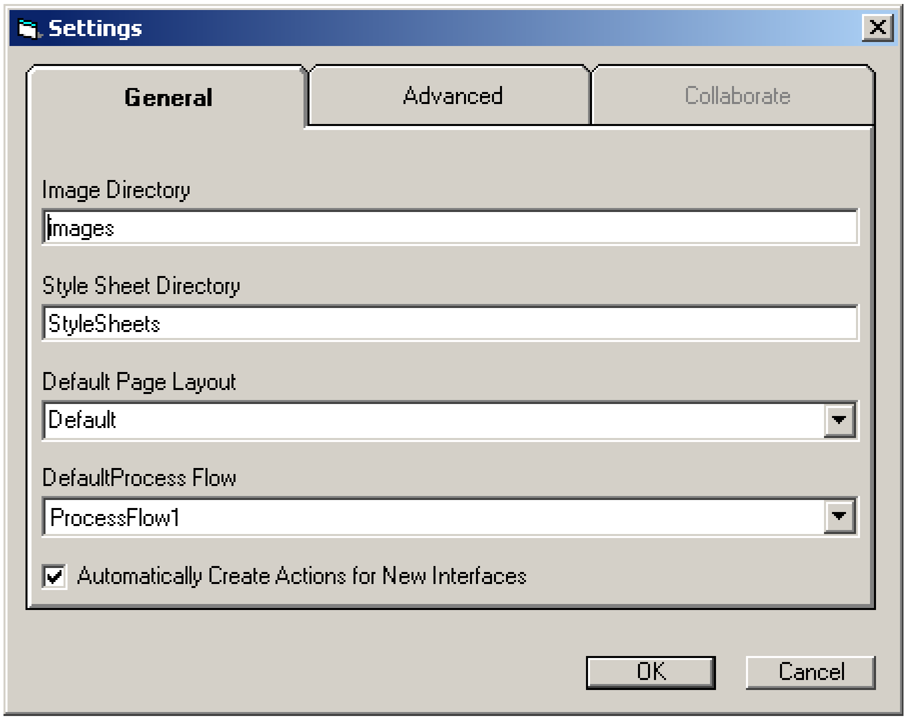


Image Directory
images

Style Sheet Directory
StyleSheets

Default Page Layout
Default

DefaultProcess Flow
ProcessFlow1

☑ Automatically Create Actions for New Interfaces

Figure 6: Settings Dialog Box - General Tab

2. Review the settings on the `General` tab of the `Settings` dialog box.

   By default, the `Image Directory` name is set to `images`, and the `Style Sheet Directory` name is set to `StyleSheets`.

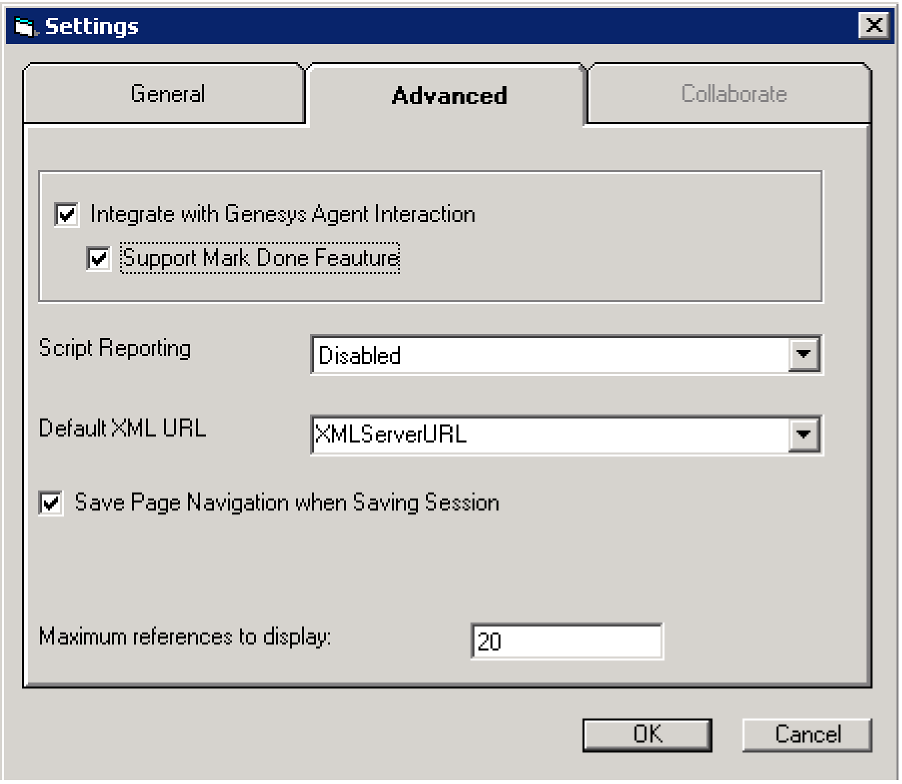
   These are the actual directories that contain the images and stylesheets that the application uses. When you compile and deploy, these directories are copied over to the target automatically.

   The `Default Page Layout` and `Default Process Flow` names are set to what you entered during the steps described in “Creating a Page Layout” on page 58 and “Creating a Process Flow” on page 56, respectively.

   The `Automatically Create Actions for New Interfaces` check box is selected by default. This indicates that Agent Scripting will automatically create a corresponding Action and insert it into the `Actions List` whenever you create a new interface.

   You may change any of these settings as desired.

3. Click the `Advanced` tab of the `Settings` dialog box as shown in Figure 7.



☑ Integrate with Genesys Agent Interaction

☑ Support Mark Done Feauture

Script Reporting: Disabled

Default XML URL: XMLServerURL

☑ Save Page Navigation when Saving Session

Maximum references to display: 20

Figure 7: Settings Dialog Box - Advanced Tab

4. Review the settings on the `Advanced` tab.

The **`Integrate with Genesys Agent Interaction` checkbox** is selected if the Project Book is initialized with the Genesys Agent Interaction Toolkit. Selecting this check box:

- Allows for integration of generated pages with a web application that can display the pages within a Tab-Frame which can result in the pages being re-launched for the same interaction. Such is the case when the generated pages are integrated with the Genesys Agent Desktop (GAD). Refer to “Overview” on page 349 of “Toolkits” which explains in more detail the integration between a user’s web application (such as GAD) and Genesys Agent Scripting generated pages.
- Adds the item “Genesys Reporting” to the Script Reporting dropdown field. Refer to “Settings” on page 375 of “Toolkits” for more details.

- Enables the menu item `Assignment Rules` from the `Compile` menu. Refer to the table called “Compile Menu - Key Options” on page 38 for more details.

The **Support Mark Done Feature checkbox** enables the Mark Done feature for Interaction Workspace for the current Project Book. When you use Agent Scripting with Interaction Workspace, an interaction must be marked as done from the Agent Scripting side, so that all activities required by the script's logic are performed.

Select this checkbox to specify that the Mark Done button is not displayed the interaction window of Interaction Workspace, for all interactions using this agent script.

When the agent tries to close the Interaction by clicking the X in the upper right corner of the Interaction Window of the Interaction Workspace, he will see a message: `Agent Scripting plug-in reports that current Script requires Marking of interaction as done from the Script side...`

If this checkbox is cleared, you will see a message at the Process Flow Assignment Rules (Deployment Settings tab) dialog box: `Mark Done Support is disabled...`

The **`Script Reporting` field** allows you enable or disable the reporting function for this Project Book. Refer to “Reporting” on page 375 for details on Script Reporting functionality.

The **`Default XML URL` field** allows you to select a default location for an XML Server. The XML Server is listed in one of the configuration file fields contained in `WWGConfig.xml`. This file is created in the deployment folder and you can edit the file.

The **`Save Page Navigation when Saving Session` check box** is selected by default. This indicates that Agent Scripting will save the page navigation (the list of Pages that the agent has clicked through) so that the `Next` and `Previous` actions will continue to follow the call flow. If unselected, the page navigation information is lost when a session is loaded.

`Maximum references to display` refers to the Visio diagrams used with Process Flows and Streams. A *reference* is an indication that a path connects to a node outside of the current diagram. This option allows you to specify how many references are allowed to appear in the diagrams.

You may change any of these settings as desired.

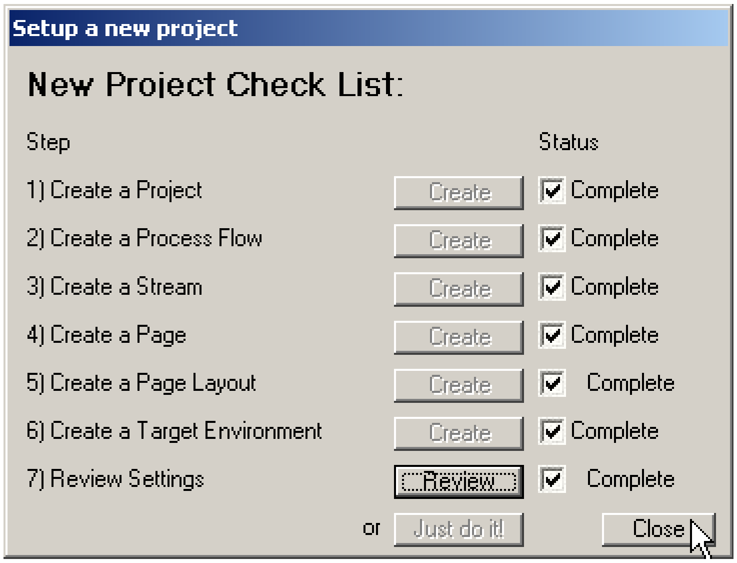
Note: Beginning with release 7.2.0, Agent Scripting uses the `POST` method only. The `Use POST Method with Forms` check box is gone. Previously, you had to specify the `POST` method, in order to prevent Agent Scripting from using the `GET` method for submitting forms.

5. Click `OK` on the `Settings` dialog box after reviewing and/or changing settings.

   The wizard (check list) `Status` check box beside `7) Review Settings` shows that this step is complete.

End of procedure

Completing New Project Book Setup



Setup a new project

New Project Check List:

Step		Status
1) Create a Project	Create	☑ Complete
2) Create a Process Flow	Create	☑ Complete
3) Create a Stream	Create	☑ Complete
4) Create a Page	Create	☑ Complete
5) Create a Page Layout	Create	☑ Complete
6) Create a Target Environment	Create	☑ Complete
7) Review Settings	Review	☑ Complete

Figure 8: New Project Book Complete

If you followed the steps in the previous sections, you will see all `Status` check boxes marked as `Complete`, as shown in Figure 8.

Procedure:
Complete New Project Book Setup

Start of procedure

1. Click `Close` to complete the setup for your new Project Book.

End of procedure

The Genesys Agent Scripting Development Environment opens. The software has created a Project, Process Flow, Stream, Page, and (Default) Page Layout, and has given each component the name you provided during setup. In addition, you have a Target Environment to which scripts will be compiled.

Genesys

Chapter

4 Creating Fields

This chapter provides an overview of the process of creating Fields and setting Field properties.

The information in this chapter is divided among the following topics:

What Is a Field?

A Field is any data value gathered or stored by Genesys Agent Scripting. This includes information that you enter into a form on a web page, or data gathered from outside applications.

Once a value has been stored in a Field, it remains in the Field even if you move on to another Page. In fact, data remains available throughout the entire session. Furthermore, you do not have to explicitly refresh the page to see updated data because this is done automatically, without any user interaction.

Note: All simple field types (excluding `Table`, `Radio Button`, `Dropdown List`, and `Custom Field Type`) can now be updated asynchronously through an Async Interface. Previously, only a Listener field could be updated by asynchronous events.

Types of Fields

There are several different types of Fields that you can create within the Genesys Agent Scripting Development Environment, including:

- String
- Multiline Text
- Numeric
- Date and Time
- Calendar
- Check Box
- Radio Button
- Dropdown List
- Table
- Custom Field Type
- Listener

Note: The Function field has been deprecated in the 7.2 release of Genesys Agent Scripting. Genesys Agent Scripting 7.2 automatically converts Function type fields to fields of type Date and Time when an older version of a Project Book is opened.

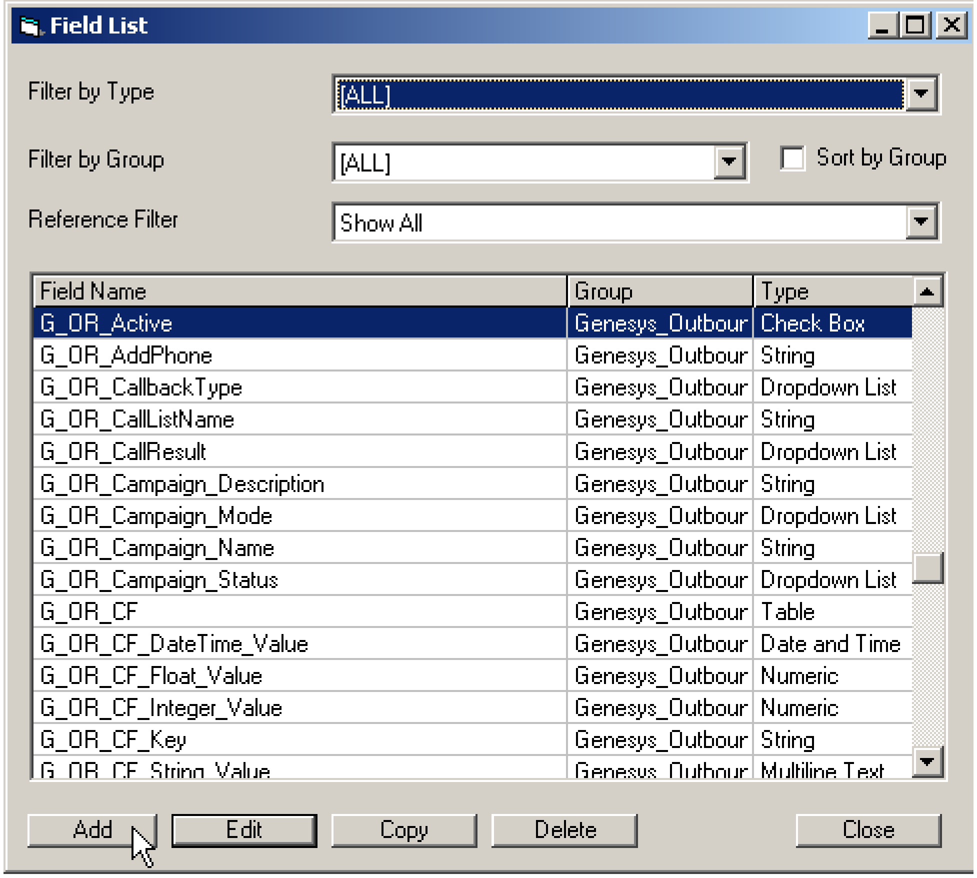
Creating a Field

Procedure: Create a Field

Purpose: Create a new Field using the Define menu.

Start of procedure

1. Select Define > Fields to open the Field List dialog box as shown in Figure 9.



Filter by Type [ALL]

Filter by Group [ALL] □ Sort by Group

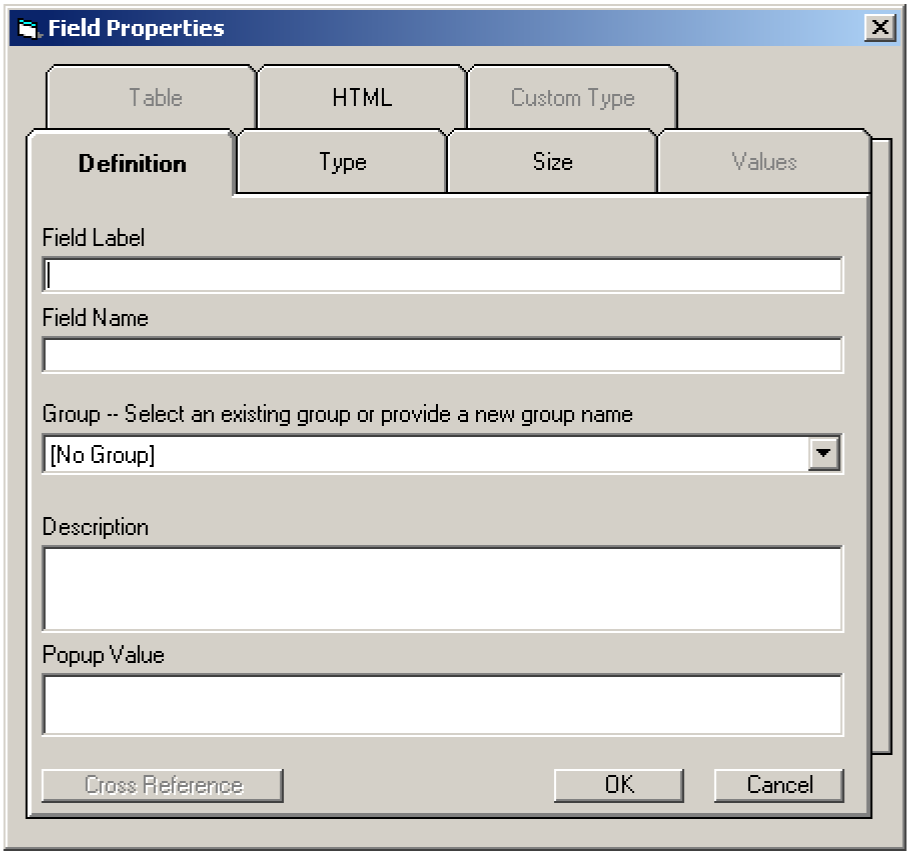
Reference Filter Show All

Field Name	Group	Type
G_OR_Active	Genesys_Outbour	Check Box
G_OR_AddPhone	Genesys_Outbour	String
G_OR_CallbackType	Genesys_Outbour	Dropdown List
G_OR_CallListName	Genesys_Outbour	String
G_OR_CallResult	Genesys_Outbour	Dropdown List
G_OR_Campaign_Description	Genesys_Outbour	String
G_OR_Campaign_Mode	Genesys_Outbour	Dropdown List
G_OR_Campaign_Name	Genesys_Outbour	String
G_OR_Campaign_Status	Genesys_Outbour	Dropdown List
G_OR_CF	Genesys_Outbour	Table
G_OR_CF_DateTime_Value	Genesys_Outbour	Date and Time
G_OR_CF_Float_Value	Genesys_Outbour	Numeric
G_OR_CF_Integer_Value	Genesys_Outbour	Numeric
G_OR_CF_Key	Genesys_Outbour	String
G_OR_CF_String_Value	Genesys_Outbour	Multiline Text

Figure 9: Field List Dialog Box

2. Click `Add` to initiate the process for creating a new Field.

   The `Field Properties` dialog box opens as shown in Figure 10.



Group -- Select an existing group or provide a new group name: [No Group]

Figure 10: Field Properties Dialog Box - Definition Tab

The `Field Properties` dialog box is where you define the characteristics that are unique to each Field you wish to create.

Note: The Function tab was removed for version 7.2.0. Genesys Agent Scripting 7.2 automatically converts `Function` type fields to fields of type `Date and Time` when an older version of a Project Book is opened.

End of procedure

Definition Tab

Table 12describes the options on the `Definition` tab of `Field Properties`.

Table 12: Field Properties, Definition Tab

Control	Description / What to Do...
Field Label	A text field that holds the value for this Field that is displayed at runtime. Labels appear on Tables and Radio Button fields. Type a label for the Field you are creating.
Field Name	A text field that holds the name that appears in and is referenced by the Agent Scripting Editor. Click this field or `Tab` to it to populate the Field with a name that corresponds to the label you just entered.
Group -- Select an existing group or provide a new group name	A dropdown list from which you can: • Select an existing `Field Group` to which this new Field will be assigned. • Define a new `Field Group` to which this new Field will be assigned. See "Field Groups". • Accept the default `[No Group]` to choose *not* to assign the new field to a `Field Group`.
Description	(Optional) A text field that holds a description for this Field. Type a description for this Field.
Popup Value	(Optional) A text field that holds a popup value for this Field. When you move the mouse over the Field at runtime, a popup appears with the value typed here. Type a popup value for this Field.

Field Groups

Genesys Agent Scripting enables you to define `Field Groups` and assign individual Fields to `Field Groups`. You create a new `Field Group` within the `Definition` tab of the `Field Properties` dialog box.

`Field Groups` are a convenient way to organize a set of Fields that share similar characteristics. For example, you might set up a `Field Group` called `Customer Data`, then assign all Fields pertaining to customer information (such as name, address, phone, e-mail, and so on) to this `Field Group`. Another `Field Group` called `Agent Data` can organize a separate set of Fields pertaining to the agent.

Procedure:Creating a Field Group

Start of procedure

To create a Field Group:

1. Select the Group dropdown list from the Definition tab of the Field Properties dialog box.
2. Select [Prompt for new value] as shown in Figure 11.

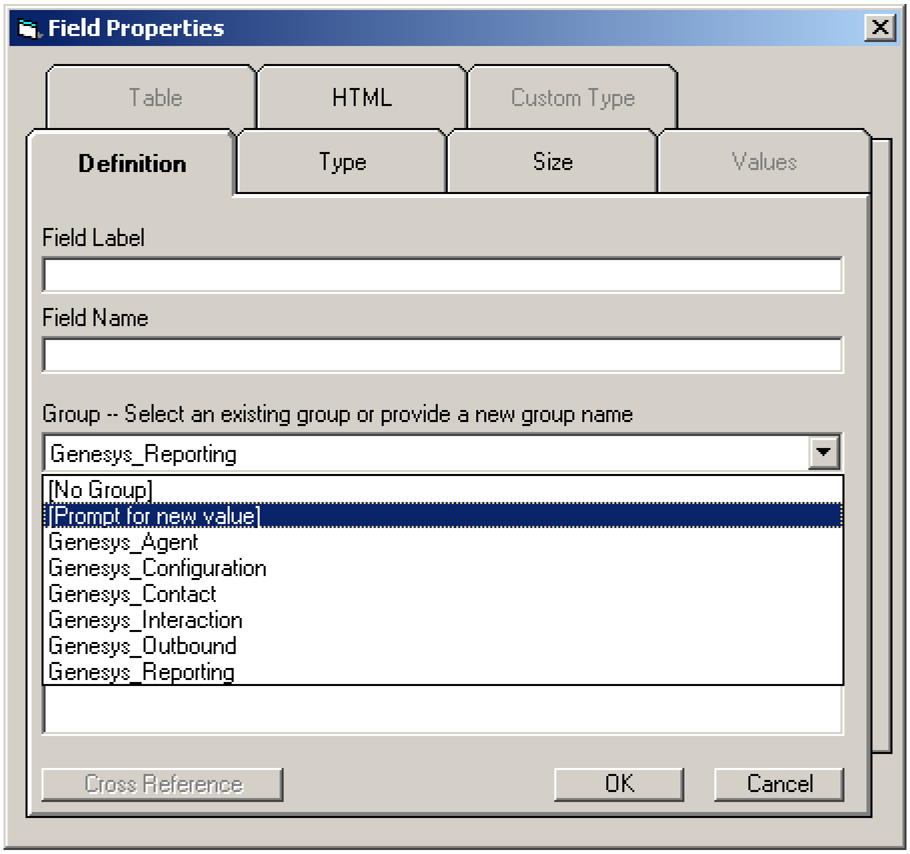
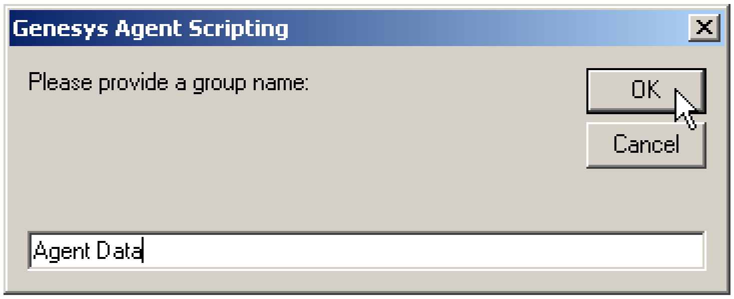


Figure 11: Create a New Field Group

A dialog box opens in which you name the new Field Group, as shown in Figure 12.



Please provide a group name:

Agent Data

Figure 12: Assign a Name to the New Field Group

3. Type a name, then click OK to assign the name to this Field Group.
4. To assign the Field you are currently defining or editing to this Field Group, click OK in the Field Properties dialog box.

End of procedure

Procedure: Assigning a Field to a Field Group

Purpose: You can assign a Field to a Field Group at the same time you are defining the Field.

Start of procedure

1. Select the **Group** dropdown list from the **Definition** tab of the **Field Properties** dialog box for the Field you are defining.
2. In the dropdown list, select the name of a previously-defined Field Group to which this Field will be assigned.
3. Finish setting up your new Field, then click OK in the Field Properties dialog box.

   The new Field will be assigned to the Field Group you selected.

End of procedure

Procedure: Assigning an Existing Field to an Existing Field Group

Purpose: You can also assign an existing (previously-defined) Field to a Field Group.

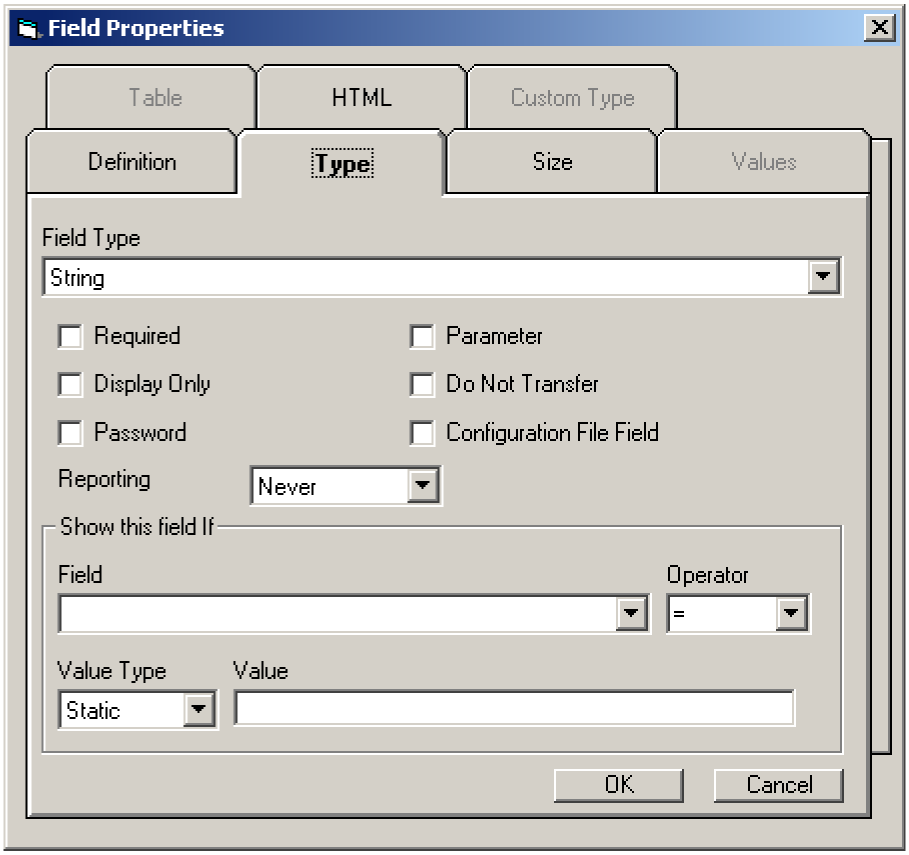
Start of procedure

1. Select the existing Field in the Field List and click Edit.
2. Select the Group dropdown list from the Definition tab of the Field Properties dialog box for the Field you are editing.
3. In the dropdown list, select the name of a previously-defined Field Group to which this Field will be assigned.
4. Click OK in the Field Properties dialog box.
5. The Field will be assigned to the Field Group you selected.

End of procedure

Type Tab

Figure 13shows the Type tab of the Field Properties dialog box.



Field Type: String

□ Required
□ Parameter
□ Display Only
□ Do Not Transfer
□ Password
□ Configuration File Field

Reporting: Never

Field:
Operator: =

Value Type: Static
Value:

Figure 13: Field Properties Dialog Box - Type Tab

Table 13 describes the options and values on the Type tab of Field Properties.

Table 13: Field Properties, Type Tab

Control	Description / What to Do...
Field Type	Select the Field type for this Field from the dropdown list. String is the default Field type. See "Descriptions of Field Types" on page 75 for more information on Field types.
Required	This check box indicates that the script user must provide a value for this Field. Genesys Agent Scripting prevents the script user from leaving the Page if a value has not been provided. Can be set for all Field types except Table.
Display Only	The script user cannot modify the content of the Field if this check box is selected. Content will appear in a script as text. This Field displays only if it has a value.
Password	This check box masks the text in the Field as a password. Can be set only for String Field types.
Parameter	This check box designates a Field that may be passed to Genesys Agent Scripting at the start of the script via the URL. Values for Parameter fields will generally be provided by applications, such as a CTI (Computer Telephony Integration) application. Parameter and Configuration File Field types are mutually exclusive. Can be set for all Field types except Table.
Do Not Transfer	This check box designates a Field whose value should not be transferred on a Save Session or Load Session action. This option is useful for data elements that should not be carried forward when a session is transferred.
Configuration File Field	This check box indicates that the Field is read from the configuration file when the Genesys Agent Scripting application is started. Parameter and Configuration File Field types are mutually exclusive.

Table 13: Field Properties, Type Tab (Continued)

Control	Description / What to Do...
Reporting	Select the type of reporting for this Field from the `Reporting` dropdown list. Table Field type is not supported. `Never`: This field is not included in Reporting Data. `If Changed`: This field is included in Reporting Data if is has changed since the last Reporting Data Collection Point (Marked Page Unload) `Always`: This field is included in Reporting Data at every Reporting Data Collection Point (Marked Page Unload). For more information refer to the “Reporting” on page 375.
Show This Field If	Fields can be shown or hidden based on the value of a specific Field. See “Making a Field Context-Sensitive” on page 97 for more information.
Field	This is the Field whose value will determine whether the Field you are currently defining will be shown or hidden. Select a Field from the `Field` dropdown list.
Operator	Depending on the Field chosen above, Genesys Agent Scripting automatically determines the valid operators and range of values. Select the appropriate operator from the dropdown list here. For example, if you select a Check Box Field type, then only the equal (`=`) and not equal (`<>`) operators will be selectable, and only `True` and `False` for values will be selectable. If you select a Radio Button Field type, only the equal (`=`) and not equal (`<>`) operators will be selectable, and the `Value` dropdown list will list all possible values for the Radio Button.
Value Type	Select either a `Field` or a `Static` value type from the dropdown list to compare your Field against. `Static` informs the application that the defined value is fixed and will not change. `Field` indicates that the value for comparison will come from an existing Field.
Value	Type or select the appropriate value in this Field.

Descriptions of Field Types

String Fields

Strings are the default Field type. `String` fields allow you to enter any sequence of characters on to a web page, such as a name, address, description, and so on.

`String` fields appear on a web page as a regular input box on a form.

Multiline Text Fields

`Multiline Text` fields allow you to enter any series of characters, just as you would for a `String` field. However, the contents of this Field can wrap around and occupy multiple lines.

`Multiline Text` fields are often used when you need a Field that will hold a paragraph of information, such as comments or special instructions.

Numeric Fields

`Numeric` fields look the same as `String` fields when displayed on the web page. However, these Fields only accept *numeric* values such as:

- Integers
- Percents
- Currency
- Fixed Decimal Point
- Floating Point

If the script user attempts to enter a non-numeric value into a `Numeric` field, an error will occur. See “Using Data Validation in Fields” on page 91.

Date and Time Fields

`Date and Time` fields also look the same as `String` fields when displayed on a web page. However, these Fields only accept *date* and *time* values.

Date and Time values are stored internally in the following format:

`yyyy-MM-dd HH:mm:ss`

For example: `2004-09-30 14:05:22`

The value can be formatted to appear on the web page in a particular manner using the `Format` field on the `Size` tab of `Field Properties`. See “Size Tab” on page 77.

If the script user attempts to enter a value into a `Date and Time` field that Agent Scripting cannot reconcile as a date or time value, an error will occur. See “Using Data Validation in Fields” on page 91.

Calendar Fields

`Calendar` fields display a calendar page for the month. The script user can select a date on that month's calendar, or navigate through the months to find a particular day.

When a date is selected on a `Calendar` field, it is stored internally the same way a `Date and Time` field is stored.

Check Box Fields

Check boxes offer you a choice such as Yes or No, On or Off, Selected or Not Selected, and so on. `Check Box` fields can have only two values: checked or unchecked. The stored value is `1` for checked, and `0` for unchecked.

Check boxes appear on a web page as box that you can click to select or clear the option.

Radio Button Fields

Radio buttons allow you to select only *one* option from a list of choices. You need space on the web page sufficient to display all the choices for the option.

A set of Radio buttons appears on a web page as a list of options, only one of which can have its Radio button selected. The arrangement of the options is flexible, but they function as a unit.

Dropdown List Fields

Dropdown Lists, like Radio buttons, also allow you to select one option from a list of choices. Use a Dropdown List when you have a large number of choices and insufficient screen space to display them all, such as cities or states.

A Dropdown List appears on a web page as a box with an arrow that you click to display the list of choices. You then highlight your selection and click to choose an option.

Table Fields

`Table` fields are collections of other Fields. A `Table` field is displayed on the web page as a table with columns made up of other Fields. A `Table` field can contain other `Table` fields to support an unlimited level of parent-child relationships.

Tables can be used to display and relate data, such as the first name, last name, phone number, and address of a customer.

Custom Field Types

A `Custom Field Type` field is a user defined Field. Custom Field Types allow you to customize Field types that are specific to your needs.

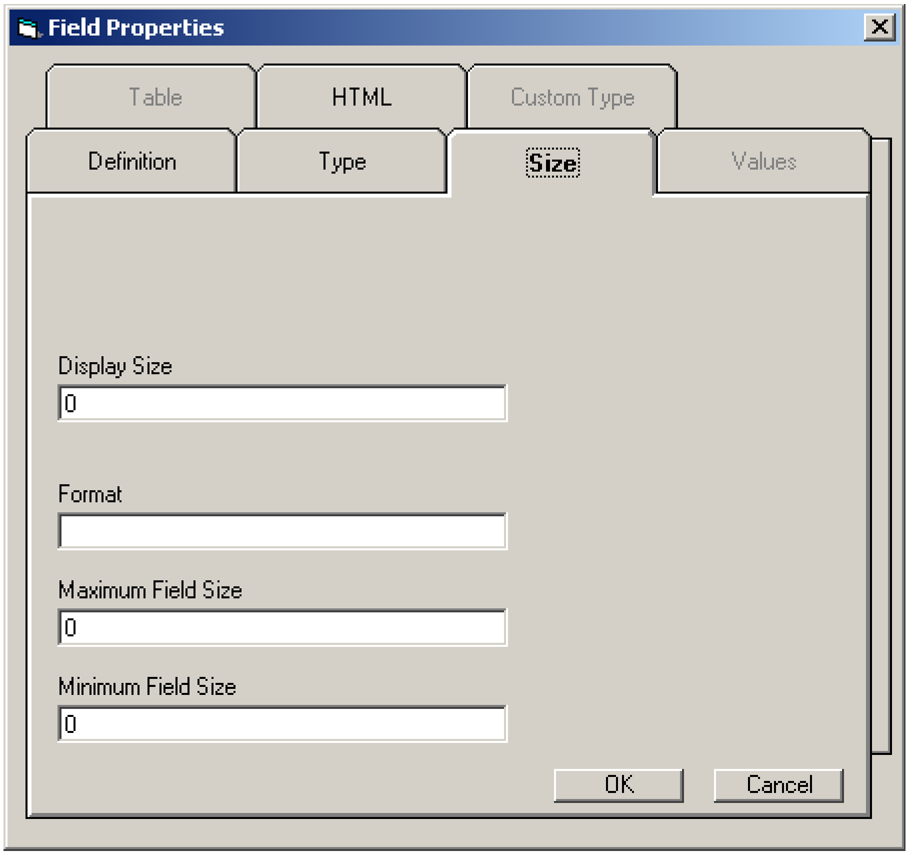
Listener Fields

A Listener field is a field type whose value can be updated asynchronously. Listener fields are display only.

Size Tab

Figure 14shows the Size tab of the Field Properties dialog box. The Size tab is available for String, MultiLine Text, Numeric, and Date and Time Field types.

If a MultiLine Text field is being defined, the Display Size field allows you to select the number of rows and columns to display, and the Format field is not provided.



Display Size
0

Format

Maximum Field Size
0

Minimum Field Size
0

Figure 14: Field Properties Dialog Box - Size Tab

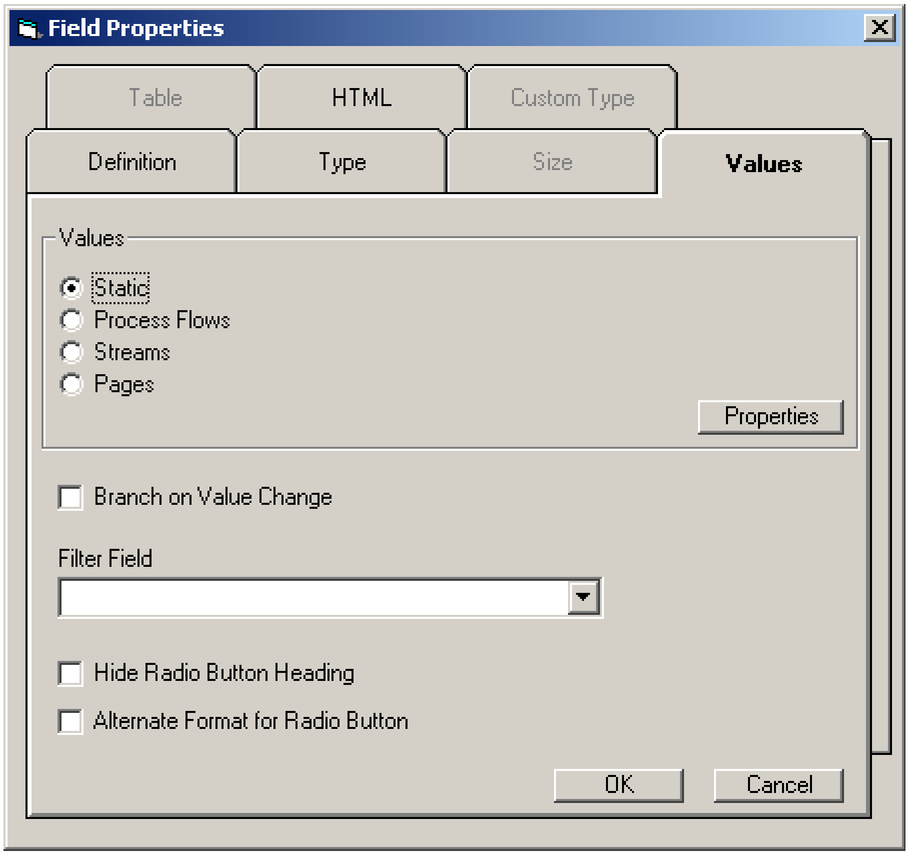
Table 14 describes the options and values on the Size tab of Field Properties.

Table 14: Field Properties, Size Tab

Control	Description / What to Do...
Display Size	Display size sets the visual size of the Field at runtime. It does not limit the number of characters that may be entered in the Field. The default value is `0,` which indicates that the Field will occupy whatever space is needed to display it. For `MultiLine Text` fields you can select the number of rows and columns to display, or keep the default (`0, 0`) to display all rows and columns.
Format	This field is where you define the format of the expected data that will be entered into the Field, and values for this field depend on the `Field Type` selected on the `Type` tab. See “Using Data Validation in Fields” on page 91 for more details on use of the `Format` field. There is no `Format` field displayed for a `MultiLine Text` field.
Maximum Field Size	This text field holds a value that indicates the maximum number of characters that the Field will hold. The default value is `0,` which indicates that the Field will have *no* maximum size. See “Maximum Field Size, Minimum Field Size” on page 96 for how this control is used for data validation.
Minimum Field Size	This text field holds a value that indicates the minimum number of characters that the Field will hold. The default value is `0,` which indicates that the Field will have *no* minimum size. See “Maximum Field Size, Minimum Field Size” on page 96 for how this control is used for data validation.

Values Tab

Figure 15 shows the `Values` tab of the `Field Properties` dialog box. The `Values` tab is only available for `Radio Button` and `Dropdown List` Field types.



Values

● Static
○ Process Flows
○ Streams
○ Pages

☐ Branch on Value Change

Filter Field

☐ Hide Radio Button Heading

☐ Alternate Format for Radio Button

Figure 15: Field Properties Dialog Box - Values Tab

Table 15 describes the options and values on the `Values` tab of `Field Properties`.

Table 15: Field Properties, Values Tab

Control	Description / What to Do...
Values	Select the `Static`, `Process Flows`, `Streams`, or `Pages` radio button for your desired value type.
Static	Static informs the application that the defined value is fixed and will not change.
Process Flows	Process Flows allows you to choose from among the existing Process Flows as selectable options for the script user in a Radio Button or Dropdown List.
Streams	Streams allows you to choose from among the existing Streams as selectable options for the script user in a Radio Button or Dropdown List.

Table 15: Field Properties, Values Tab (Continued)

Control	Description / What to Do...
Pages	Pages allows you to choose from among the existing Pages as selectable options for the script user in a Radio Button or Dropdown List.
Properties	Click Properties, then click Yes to save the Field if a message appears. The Field Value List dialog box opens. See the next section, "Adding Values" for more information.
Branch on Value Change	Select this check box to have Genesys Agent Scripting automatically branch each time the value of the Field changes.
Filter Field	You can select a Field to act as a *filter* for this Radio Button field or Dropdown List field. When you select a Field to be a filter from the dropdown list, its value will be used to determine the values that will be displayed on the script page.
Hide Radio Button Heading	Select this check box if you want the Radio Button heading not to appear on the window at runtime. This option is disabled for Dropdown List fields.
Alternate Format for Radio Button	This check box allows Radio Buttons to be displayed horizontally without a border and without a title. This option is disabled for Dropdown List fields.

Adding Values

The Properties button on the Field Properties dialog box allows you to add values to Fields. Click Add on the Field Value List dialog box shown in Figure 16 to initiate the process for assigning values to this Radio Button or Dropdown List, and to open the Field Value dialog box shown inFigure 17.

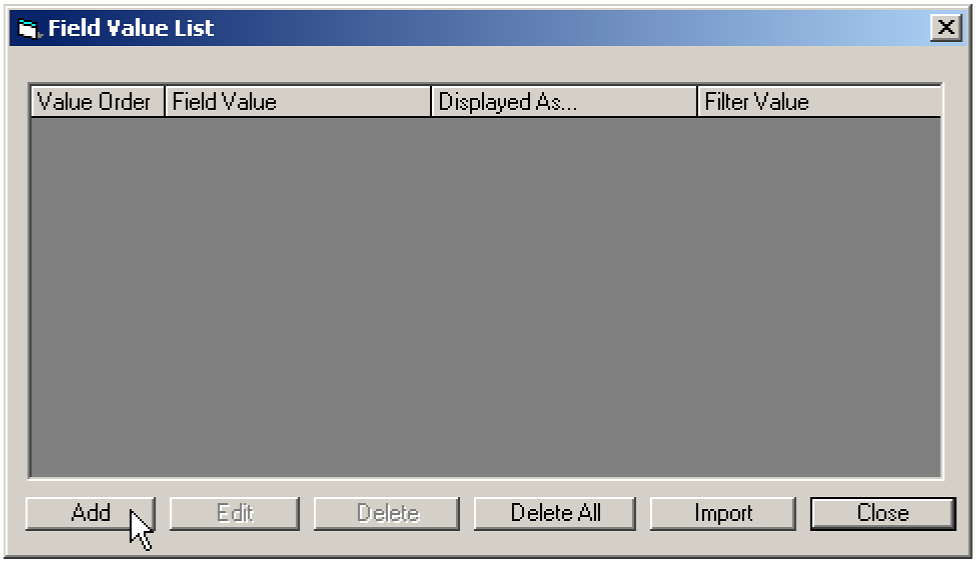
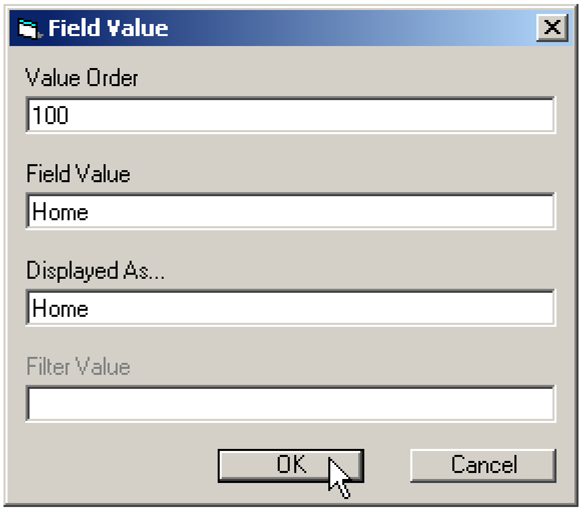


Figure 16: Field Value List Dialog Box



Value Order: 100

Field Value: Home

Displayed As...: Home

Filter Value:

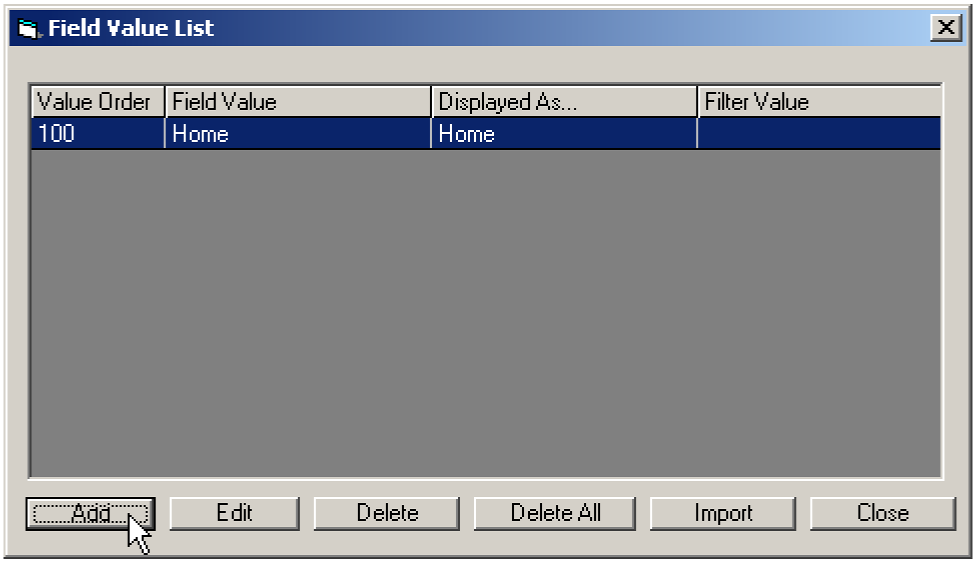
Figure 17: Field Value Dialog Box

Table 16 describes the fields on the `Field Value` dialog box.

Table 16: Field Value Dialog Box

Control	Description / What to Do...
Value Order	Value Order is a numeric field that allows you to order the items in the Radio Button or Dropdown List (smallest to largest).
Field Value	Type a Field Value, which represents the name that will be displayed for this Field on the web page.
Displayed As...	Displayed As... defines how the selection is referenced from within Agent Scripting. Click this field or Tab to it to populate the Field with a name that corresponds to the Field Value you previously entered, or type another Displayed As... value.
Filter Value	Filter Value is the value assigned to this value for filtering. Filtering can be used to reduce the number of choices the person using the scripts has to look at.
OK	Click OK to finish adding a value for this Field.

Your new Radio Button or Dropdown List value is added to the Field Value List dialog box. An example with a Radio Button value called Home is shown in Figure 18.



Value Order	Field Value	Displayed As...	Filter Value
100	Home	Home	

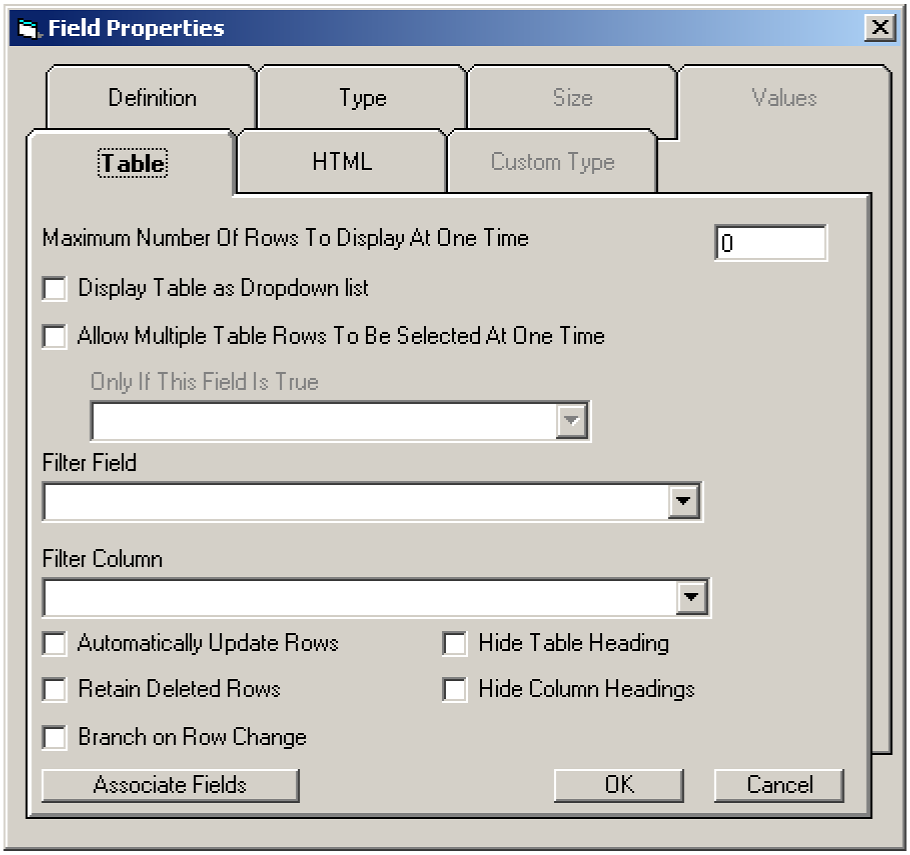
Figure 18: Field Value List - Home Added

Click Add on the Field Value List dialog box to add more values for this Field and repeat the steps above, or

Click Close when you have finished adding all values to this Field.

Table Tab

Figure 19shows the Table tab of the Field Properties dialog box. The Table tab is only available for the Table Field type.



Maximum Number Of Rows To Display At One Time 0

□ Display Table as Dropdown list

□ Allow Multiple Table Rows To Be Selected At One Time

Only If This Field Is True

Filter Field

Filter Column

□ Automatically Update Rows □ Hide Table Heading

□ Retain Deleted Rows □ Hide Column Headings

□ Branch on Row Change

Figure 19: Field Properties Dialog Box - Table Tab

Table 17 describes the options and values on the Table tab of Field Properties.

Table 17: Field Properties, Tables Tab

Control	Description / What to Do...
Maximum Number of Rows to Display at One Time	This setting indicates the maximum number of rows that Genesys Agent Scripting should display on the script page in the browser. If the number of rows in the table exceeds this number, Genesys Agent Scripting will automatically add First, Previous x, Next x, and Last buttons to the script page where x is equal to the display rows.
Display Table as Dropdown List	When this check box is selected, the table is displayed as a dropdown list containing the column headings.
Allow Multiple Table Rows to be Selected at One Time	Select this check box to indicate that multiple rows in a table can be selected at one time. Each row will have a Check Box in the first column so that the script user can select more than one row. By default, Genesys Agent Scripting tables are displayed with a Radio Button in the first column of each row so a user can select only one row.
Only if This Field is True	This dropdown list field becomes active if the `Allow Multiple Table Rows to be Selected at One Time` field is selected. This item allows you to set a condition on whether or not multiple rows in a table can be selected. Only `Check Box` fields will appear in this list.
Filter Field	Select a `Filter` field to be used to evaluate each row in the table. If the Filter column in the field matches the value in `Filter Field`, the row will be displayed; otherwise the row will be hidden.
Filter Column	Selects the column within the table whose value will be matched against the value of `Filter Field` to determine if the row will be displayed.
Automatically Update Rows	Select this check box to indicate if table values should automatically be updated. A table is a collection of Genesys Agent Scripting Fields. If auto update is activated, values in Fields are automatically copied to rows in a Genesys Agent Scripting table. If auto update is not activated, you must press a button to have changes copied into a row in a table.

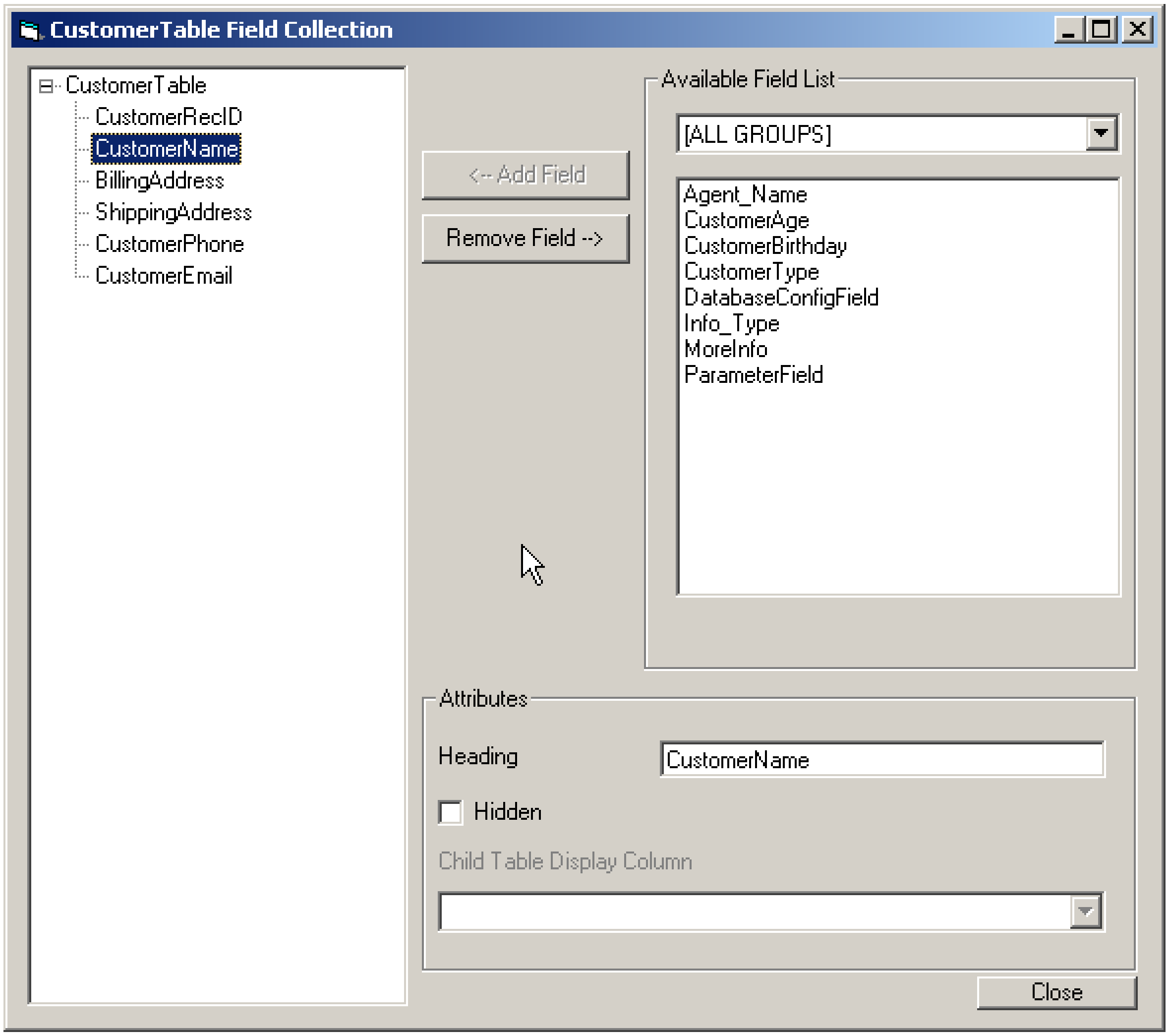
Table 17: Field Properties, Tables Tab (Continued)

Control	Description / What to Do...
Retain Deleted Rows	When a `Delete Row` command is issued, the row will be marked for deletion and hidden from the user if you select this check box. The row will not actually be removed from the table. If this option is not selected, a `Delete Row` command will cause the row to be removed from the table.
Branch on Row Change	Select this check box for Genesys Agent Scripting to automatically generate a branch when a row in the table is selected.
Hide Table Heading	Select this check box for the table's heading *not* to appear on the web page at runtime.
Hide Column Headings	Select this check box to display the table's columns on the web page at runtime *without* the column headings.
Associate Fields	Click `Associate Fields` to open a `Field Collection` dialog box for the Table Field, which enables you to associate (or map) already-defined Fields to columns of the table.

Associating Fields with the Table

An example of a `Field Collection` dialog box for a Table Field is shown in Figure 20.

This dialog box displays tables in a tree structure that allows you to maintain the current table as well as any sub-tables. You can drag and drop Fields onto the table, as well as reorder Fields. The Available Field list is managed to ensure that you do not add the same Field to the structure twice. Changes to the attributes are automatically saved.



Heading CustomerName

☐ Hidden

Figure 20: Field Collection Dialog Box

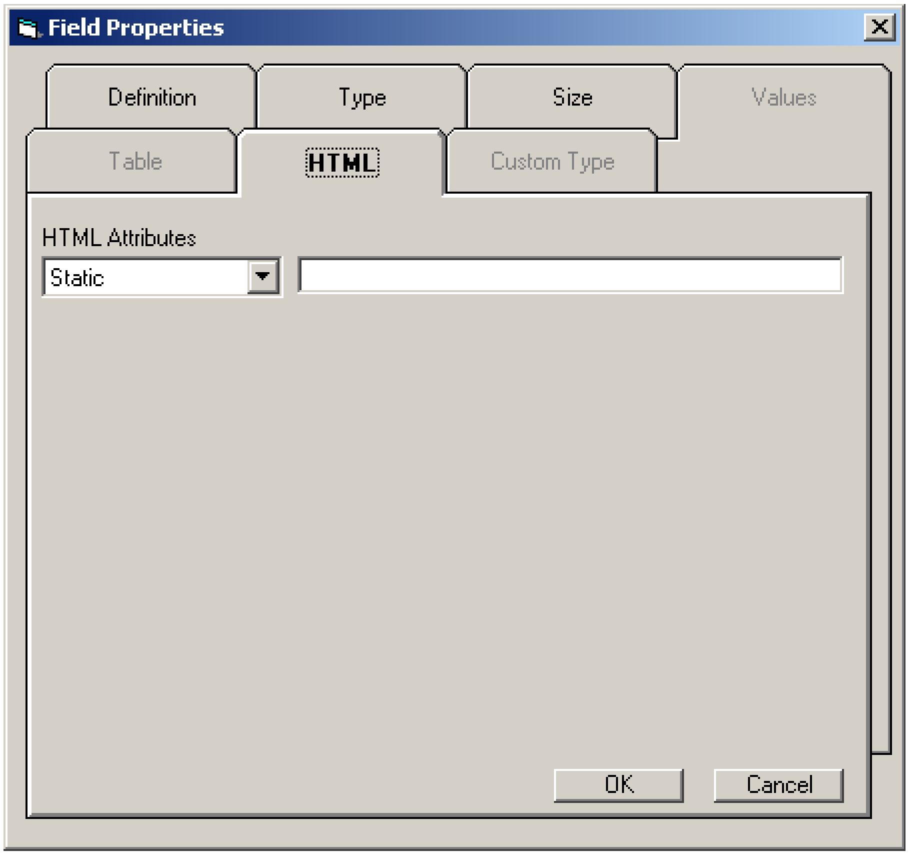
Table 18 describes the options and values on the `Field Collection` dialog box.

Table 18: Associate Fields Dialog Box

Control	Description / What to Do...
Table Tree	This box displays the Fields that belong to the table. Reorder the Fields in the tree structure by dragging and dropping. The columns appear in the table from left to right in the same order that the Fields appear in the tree, from top down.
Available Field List	This list contains all the Fields that can be added to the table. Once a Field is added, its name is removed from the list. Select a `Field Group` from the `Available Field List` dropdown list to show only those Fields that belong to the selected `Field Group`.
Add Field	After selecting a Field in the `Available Field List`, click `Add Field` to add the Field to the table.
Remove Field	After selecting a Field in the `Table Tree` box, click `Remove Field` to remove the Field from the table.
Heading	Type the text to be displayed as a column heading in the table.
Hidden	Select this check box to hide the values of the Field so that the script user cannot see them when this table is displayed. This option is helpful if you need to include a Field in a table but do not wish to display its values on the web page.
Child Table Display Column	If the Field you selected above is a `Table` field, you can select a Field from that table and it will be used for this column in the new `Table` field.
Close	Click `Close` to close the dialog box.

HTML Tab

Figure 21shows the HTML tab of the Field Properties dialog box. The HTML tab is available for all fields except Custom Field and Listener Field types, as long as the Display Only check box on the Type tab is not selected.



HTML Attributes: Static

Figure 21: Field Properties Dialog Box - HTML Tab

Table 19 describes the options and values on the HTML tab of Field Properties.

This tab is used to change the look and feel of the Field such as font, color, and spacing. It is a way for the developer to easily add custom HTML tags associated with this Field.

Table 19: Field Properties, HTML Tab

Control	Description / What to Do...
HTML Attributes	Select either a `Field` or a `Static` value type from the HTML Attributes dropdown list. `Static` informs the application that the defined value is fixed and will not change. `Field` indicates that the value will come from an existing Field.
TR Attributes	Only applicable for Radio Button and Table field types. TR is a table row HTML tag. Select either a `Field` or a `Static` value type from the TR Attributes drop-down list.
TD Attributes	Only applicable for Radio Button and Table field types. TD is a table column HTML tag. Select either a `Field` or a `Static` value type from the TD Attributes drop-down list.
TD Attributes for Current Row	Only applicable for the Table field type. The attribute will be applied to the selected row in a table. Select either a `Field` or a `Static` value type from the TD Attributes for Current Row drop-down list.
<unlabeled fields>	Type or select the appropriate value in this Field for the corresponding attribute.

Custom Type Tab

Figure 22shows the `Custom Type` tab of the `Field Properties` dialog box. The `Custom Type` tab is only available for Fields of type `Custom Field Type`.

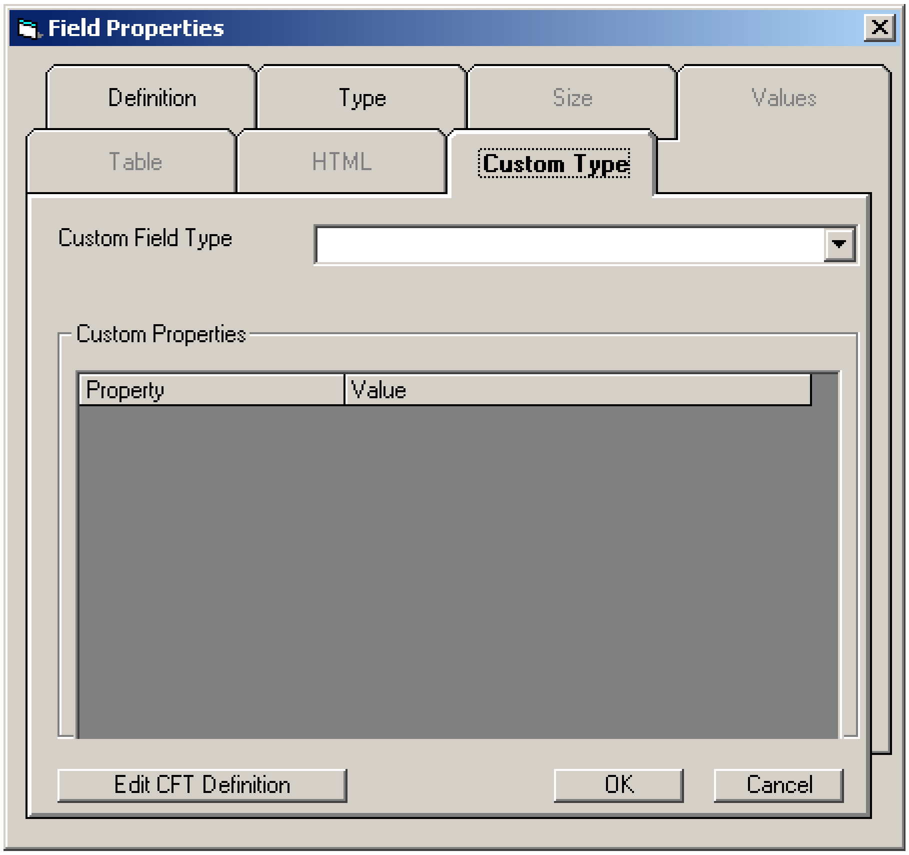


Figure 22: Field Properties Dialog Box - Custom Type Tab

Table 20 describes the options and values on the `Custom Type` tab of `Field Properties`.

Table 20: Field Properties, Custom Type Tab

Control	Description / What to Do...
Custom Field Type	Select from among the previously-defined Custom Field Types. The field you are defining will be assigned to this type. You must have already defined at least one Custom Field Type before using this dialog box.
Custom Properties	Displays the available `Properties` and `Values`.
Property	Property...

Table 20: Field Properties, Custom Type Tab (Continued)

Control	Description / What to Do...
Value	Value...
Edit CFT Definition	Opens the Custom-Field-Type Properties dialog box which allows you to edit your Custom-Field-Type selected for this Field.

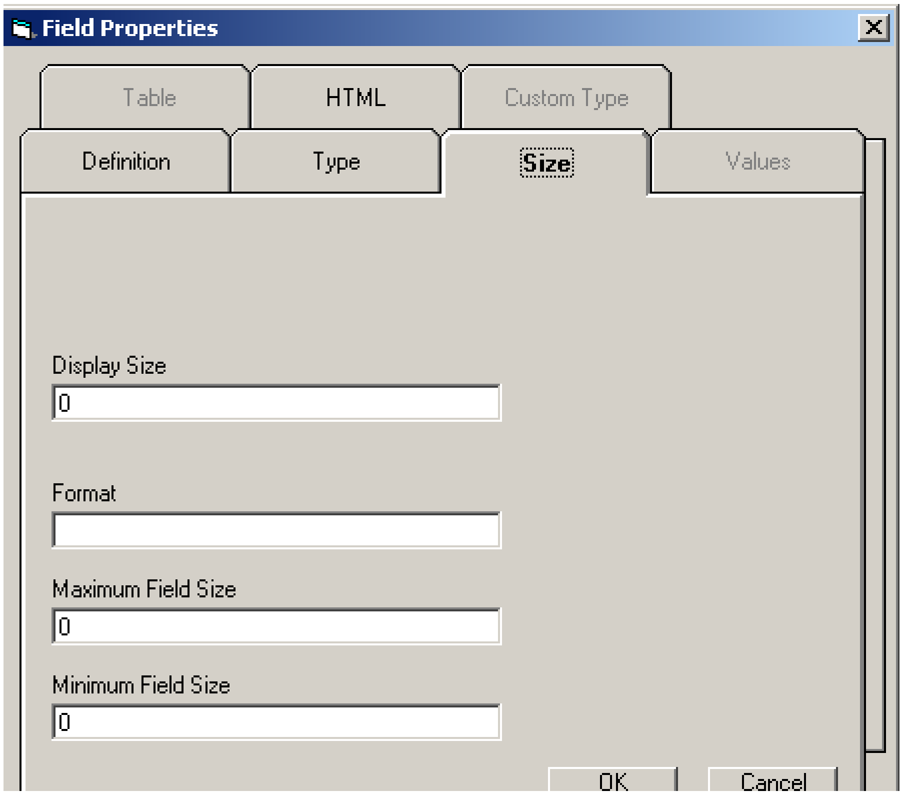
Completing the Field

Click `OK` on the `Field Properties` dialog box to complete your Field definition.

Your new Field is added to the `Field List`.

Using Data Validation in Fields

Data Validation refers to the concept of checking to see if the script user has entered valid data into a Field. Agent Scripting performs a level of data validation through the use of the `Format` field on the `Size` tab of the `Field Properties` dialog box, as shown in Figure 23.



Display Size
0

Format

Maximum Field Size
0

Minimum Field Size
0

Figure 23: Field Properties Dialog Box - Size Tab

Note: The Format field on the Size tab is only available for String, Numeric, and Date and Time Field types. Data validation, therefore, is only applicable to these Field types.

The Format Field

Values that you can enter or select for the Format field depend on the Field type you are setting up. Table 21 shows how the Format field is populated depending on the type of Field selected on the Type tab.

Table 21: Format Field Selections for Various Field Types

Field Type	Selections for Format Field on Size Tab
String	Any character can be included in a format with the # and @ characters as placeholders for user-provided data. # is used to indicate any valid number. @ is used to indicate a valid number but has the special property of being hidden. This is useful for Fields that contain sensitive information such as a credit card. See "Format Examples - String Field" on page 93.

Table 21: Format Field Selections for Various Field Types (Continued)

Field Type	Selections for Format Field on Size Tab
Numeric	For `Numeric` fields, the `Format` field becomes a dropdown list with the following selections: • Integer • Currency • Fixed Decimal Point • Floating Decimal Point • Percentage.
Date and Time	The format of a `Date and Time` field has two purposes. First, it is used to ensure that the value entered in the Field is a valid value. Secondly, it is used to tell the server how to convert the input to the standard format in which it is stored. All date and time values are stored internally in the format `yyyy-MM-dd HH:mm:ss`, where `HH:mm:ss` are optional values with the caveat that items to the right can be dropped. That is, the user can specify `HH:mm` without the `ss` value, but the user cannot specify `mm` without also specifying `HH`. See “Format Examples - Date and Time Field” on page 95.

Format Examples - String Field

Usually, a `String` field will require no special formatting value unless it is meant to hold some *numeric* information. For example, a telephone number, a zip code, and a credit card number are types of data that would most likely occupy `String` fields. Since you would not normally perform calculations on this type of information, a `Numeric` field might not be the best choice. In addition, telephone numbers, nine-digit zip codes, and credit card numbers usually include some non-numeric characters which you might want to display in the Field to the script user.

Telephone Number Format

To collect a telephone number with area code and extension.

Type the following into the `Format` field:

```
(###) ###-#### ext. ####
```

This formats the input for the script user so that the numbers entered replace the `#` sequentially.

If the script user types the following into the Field during runtime:

`92555512121000`

the number is displayed as follows in the Field:

`(925) 555-1212 ext. 1000`

In this example, typing anything other than a number would represent invalid data for this Field. The parentheses, hyphen, and `ext.` are provided automatically.

Nine-Digit Zip Code Format

To collect a nine-digit zip code.

Type the following into the `Format` field:

`#####-####`

Again, this formats the input for the script user so that the numbers entered replace the `#` sequentially.

If the script user types the following into the Field during runtime:

`123456789`

the number is displayed as follows in the Field:

`12345-6789`

The same format works if the script user types only the usual five digits of the zip code:

If the user types `12345` the number is displayed exactly as typed.

In this example, typing anything other than a number would represent invalid data for this Field. The hyphen is provided automatically for more than five digits.

Credit Card Number Format

To collect a credit card number, since this is sensitive information, you will hide all but the last four digits of the card number using the `@` special character.

Type the following into the `Format` field:

`@@@@-@@@@-@@@@-####`

This formats the input for the script user so that the numbers entered replace the `@` and `#` sequentially.

If the script user types the following into the Field during runtime:

`1234567890123456`

the number is displayed as follows in the Field:

`****-****-****-3456`

In this example, typing anything other than a number would represent invalid data for this Field. The hyphen is provided automatically.

Format Examples - Date and Time Field

Table 22 shows how date and time data is constructed for use in a Date and Time Field.

Table 22: Date and Time Field Formats

Element	Meaning	Example(s) of Use
M MM MMM MMMM	1- or 2-digit month 2-digit month abbreviated month name month name	1 01 Jan January
d dd	1- or 2-digit date 2-digit date	2 02
y yy yyyy	2- or 4-digit year 2-digit year 4-digit year	04, 2004 04 2004
E EEEE	abbreviated day name day name	Fri Friday
H HH	1- or 2- digit hour (0-23) 2-digit hour (0-23)	8 9 10 11 12 13... 08 09 10 11 12 13...
h hh	1- or 2- digit hour (1-12) 2-digit hour (1-12)	...8 9 10 11 12 ...08 09 10 11 12
m mm	1- or 2- digit minute 2-digit minute	...9 10 11 12 13 14...59 ...09 10 11 12 13 14...59
s ss	1- or 2-digit second 2-digit second	...9 10 11 12 13 14...59 ...09 10 11 12 13 14...59
a	AM (or PM)	AM

Table 23 shows examples of valid entries for the `Format` field when setting up `Date and Time` fields, as well as what is displayed to the script user and how the data is stored internally.

Table 23: Date and Time Field Examples

Format	Display	Value Stored Internally
M/d/y	1/2/04	2004-01-02 00:00:00
MM/dd/yy	01/02/04	2004-01-02 00:00:00
MM/dd/yy	01/02/99	2099-01-02 00:00:00
MMM/d/yyyy	Dec/3/2004	2004-12-03 00:00:00
E MMMM/d/yyyy	Fri December/31/1999	1999-12-31 00:00:00
M/d/y EEEE H	1/2/04 Friday 23	2004-01-02 23:00:00
M/d/y hh:mm a	12/31/1999 11:59 PM	1999-12-31 23:59:00
h:m:s a	11:59:59 PM	1970-01-01 23:59:59

Use the yyyy format if you need to display years 19xx.

Maximum Field Size, Minimum Field Size

The Maximum Field Size and Minimum Field Size fields on the Size tab also offer a level of data validation because they constrain the allowable field size.

For example, if you are setting up a String field that will hold password data, you may choose to set the Minimum Field Size to a value of 5. Then, Agent Scripting will not accept a value for this Field from the script user if that value is under five characters in length.

Likewise, if you are setting up a Numeric field that will hold an integer value for an account number, and all valid account numbers are 10 digits, you may choose to set the Maximum Field Size to a value of 10. Then, Agent Scripting will not accept a value for this Field from the script user if that value exceeds 10 digits in length.

Field Type Validation

Finally, the Field type you select also provides a level of data validation. For example, Numeric fields will only accept numeric data, and the expected data follows the format you selected in the Format field. If the script user types anything other than a number into a Numeric field, this is considered invalid data.

Making a Field Context-Sensitive

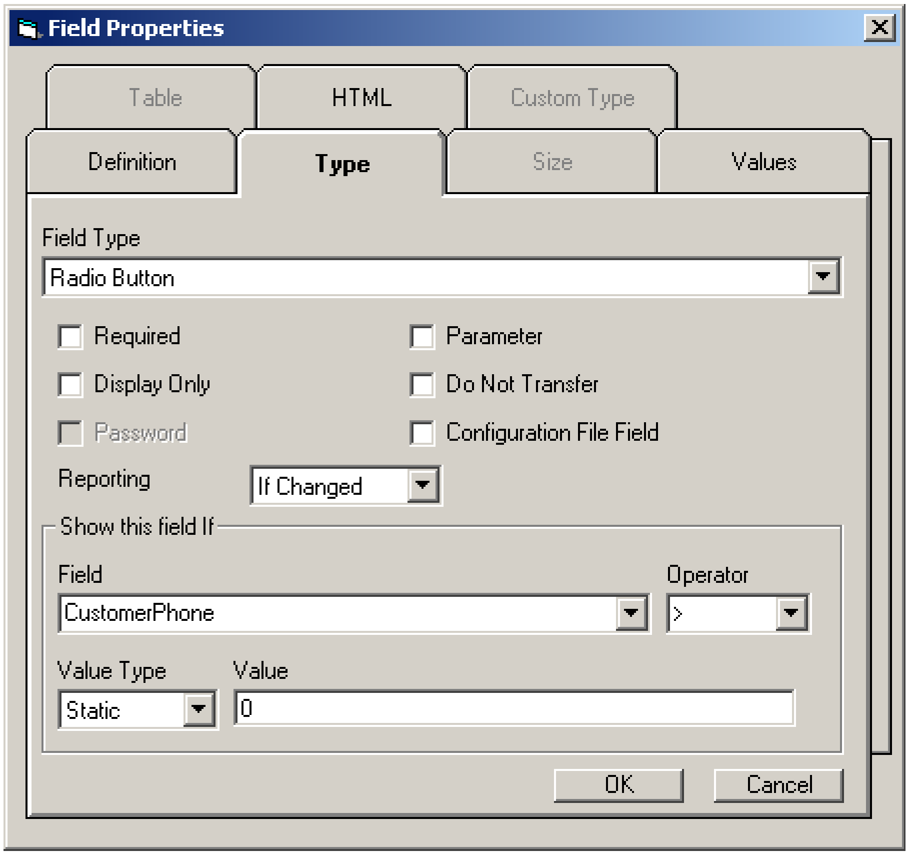
Procedure: Making a Field Context-Sensitive

Purpose: Genesys Agent Scripting allows you to make a Field *context-sensitive*. For example, assume there is a CustPhoneType field of type Radio Button in an earlier section with values Home, Work, and Mobile. Assume we have also created a CustomerPhone field of type Numeric which holds a customer's telephone number.

Let's suppose that for some customers, we will not have access to a telephone number, so the CustomerPhone field will remain blank. In these cases, there is no need to display the CustPhoneType Radio Button field on the web page. Agent Scripting allows you to set the CustPhoneType Radio Button field as a context-sensitive Field such that the Field will only display if there is a value in the CustomerPhone field.You do this from the Type tab on the Field Properties dialog box for the CustPhoneType Radio Button field.

Start of procedure

1. Select Define > Fields to open the Field List dialog box.
2. Select the CustPhoneType Radio Button field from the list and click Edit.
3. Select the Type tab on the Field Properties dialog box as shown in Figure 24.



Field Type: Radio Button

☐ Required ☐ Parameter

☐ Display Only ☐ Do Not Transfer

☐ Password ☐ Configuration File Field

Reporting: If Changed

Show this field If

Field: CustomerPhone Operator: >

Value Type: Static Value: 0

Figure 24: Making CustPhoneType Context Sensitive

4. In the Show This Field If area, select CustomerPhone from the Field dropdown list.
5. Select the “greater-than” > operator from the Operator dropdown list.
6. Select Static from the Value Type dropdown list.
7. Type a value of 0 in the Value field.

   This statement now reads, essentially:

   "Show this field if the value of the CustomerPhone field is greater than zero (0)."
8. Click OK to save the changes to the CustPhoneType field.
9. Click Close to close the Field List dialog box.

End of procedure

Genesys Agent Scripting allows a large amount of flexibility when setting up context-sensitive Fields. You must, however, have already defined and created the Field on which the context-sensitive Field depends, since, as shown in Step 4 above, you select the Field from a dropdown list.

Examples

This section shows three step-by-step examples that illustrate how to create a Field:

- Creating a String Field
- Creating a Radio Button Field
- Creating a Dropdown List Field

Creating a String Field

This field will hold data for a customer's name.

Procedure: Create a String Field

Start of procedure

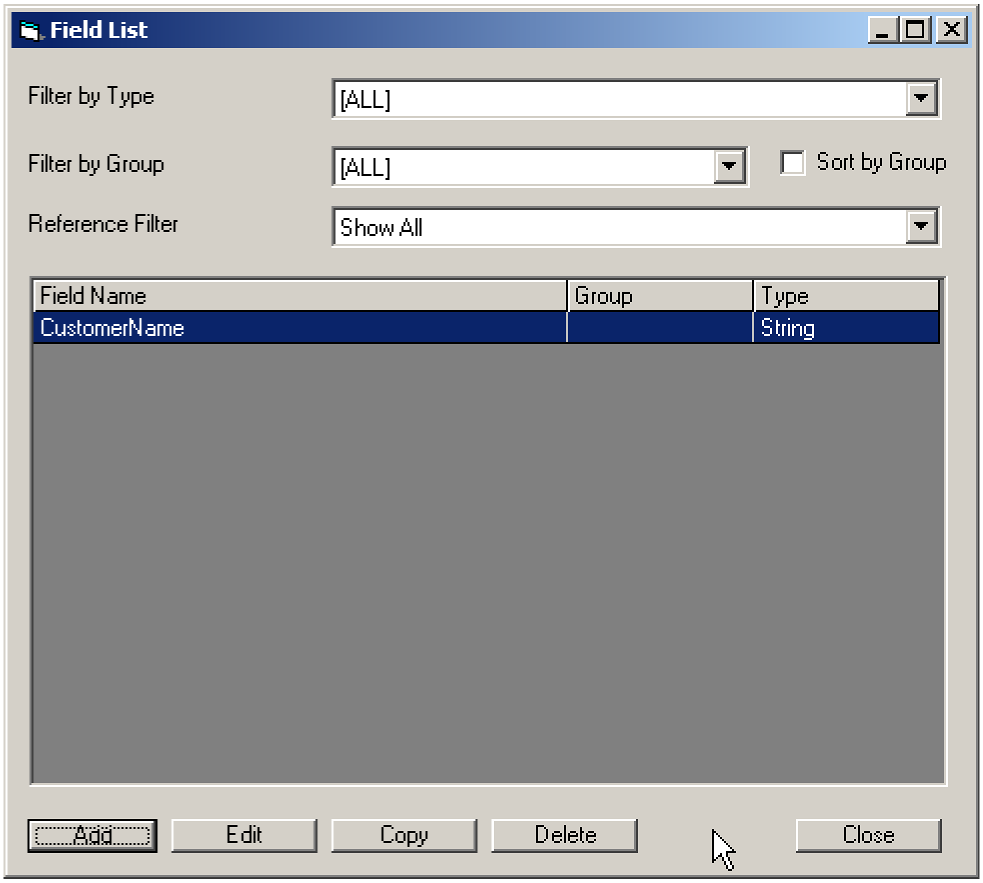
1. In the `Field Label` field on the `Field Properties` dialog box, type a label for the Field you are creating.
2. Click the `Field Name` field or `Tab` to it to populate this field with a name that corresponds to the label you just entered.
3. Click the `Type` tab on the `Field Properties` dialog box to open the `Type` tab.
4. Select Field Type `String` from the dropdown list (which is the default Field type).
5. No other options on the `Type` tab apply to our current example, so click `OK` to finish creating this Field.

End of procedure

Note: See "Making a Field Context-Sensitive" on page 97 for information on using options in the `Show This Field If` area on the `Type` tab.

Next Steps

Your new Field is added to the `Field List` as shown in Figure 25, which is normally displayed in the Genesys Agent Scripting Development Environment main window. An example `String` field called `CustomerName` is shown.



Filter by Type: [ALL]

Filter by Group: [ALL] □ Sort by Group

Reference Filter: Show All

Field Name	Group	Type
CustomerName		String

Figure 25: Field List

Creating a Radio Button Field

This field will specify whether the customer's phone number is for Home, Work, or Mobile.

Procedure:
Create a Radio Button Field

Start of procedure

1. Click `Add` on the `Field List` dialog box to initiate the process for creating a new Field and to open the `Field Properties` dialog box.
2. In the `Field Label` field on the `Field Properties` dialog box, type a label for the Field you are creating.
3. Click the `Field Name` field or `Tab` to it to populate this field with a name that corresponds to the label you just entered.
4. Click the `Type` tab on the `Field Properties` dialog box to open the `Type` tab.

5. Select Field Type `Radio Button` from the dropdown list.
6. Click the `Values` tab on the `Field Properties` dialog box to specify values for this Radio Button set.
7. In the `Values` area, select `Static`.

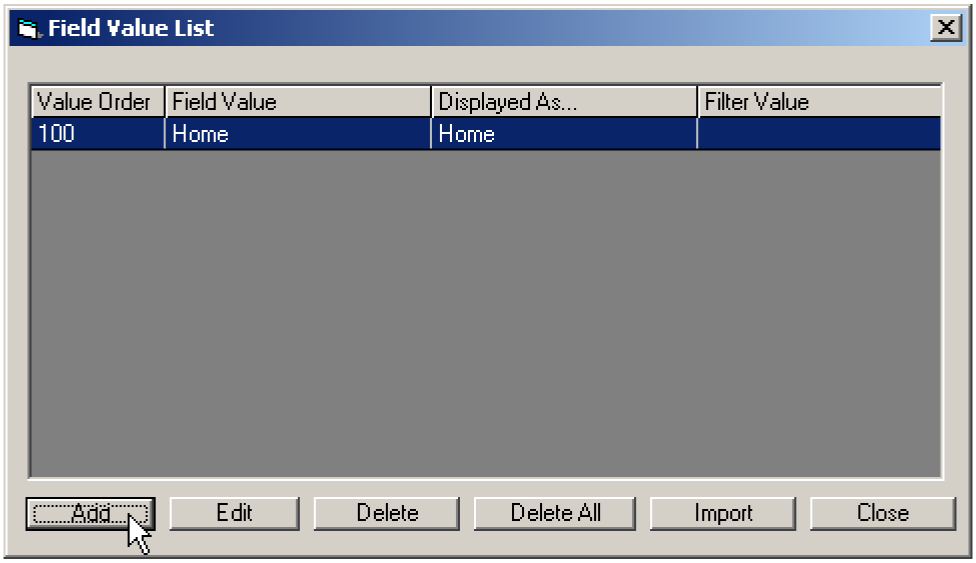
   Static informs the application that the value will not change.
8. Click `Properties`, then click `Yes` to save the Field if a message appears.
9. Click `Add` on the `Field Value List` dialog box to initiate the process for assigning values to this Radio Button, and to open the `Field Value` dialog box.
10. In the `Field Value` field on the `Field Value` dialog box, type a display value for the Radio Button option you are creating (such as `Home`).
11. Click the `Displayed As...` field or `Tab` to it to populate this field with a name that corresponds to the display value you just entered.

    The `Value Order` allows you to order the items in the Radio Button (smallest to largest).

    The `Field Value` holds the name that will be displayed for the option on the web page.

    The `Displayed As...` defines how the selection is referenced from within Agent Scripting.
12. Click `OK`.

    Your new Radio Button value is added to the `Field Value List` as shown in Figure 26.



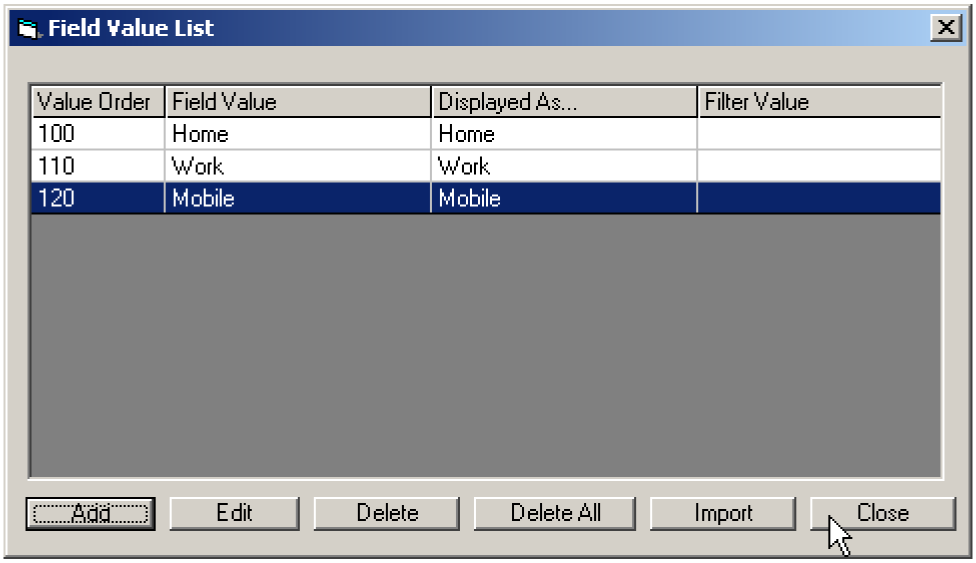
Field Value List

Value Order	Field Value	Displayed As...	Filter Value
100	Home	Home	

Figure 26: Field Value List

13. You need to add two additional values for this Radio Button. Repeat Steps 9 through 12 first using Work as the Display Value and Database Value, then using Mobile.

    The Field Value List appears as shown in Figure 27.



Value Order	Field Value	Displayed As...	Filter Value
100	Home	Home	
110	Work	Work	
120	Mobile	Mobile	

Figure 27: Field Value List - Radio Button Values Added

14. Click Close on the Field Value List after all three values are added.
15. Click OK on the Field Properties dialog box to complete your Radio Button.

End of procedure

Your new Field is added to the Field List.

Creating a Dropdown List Field

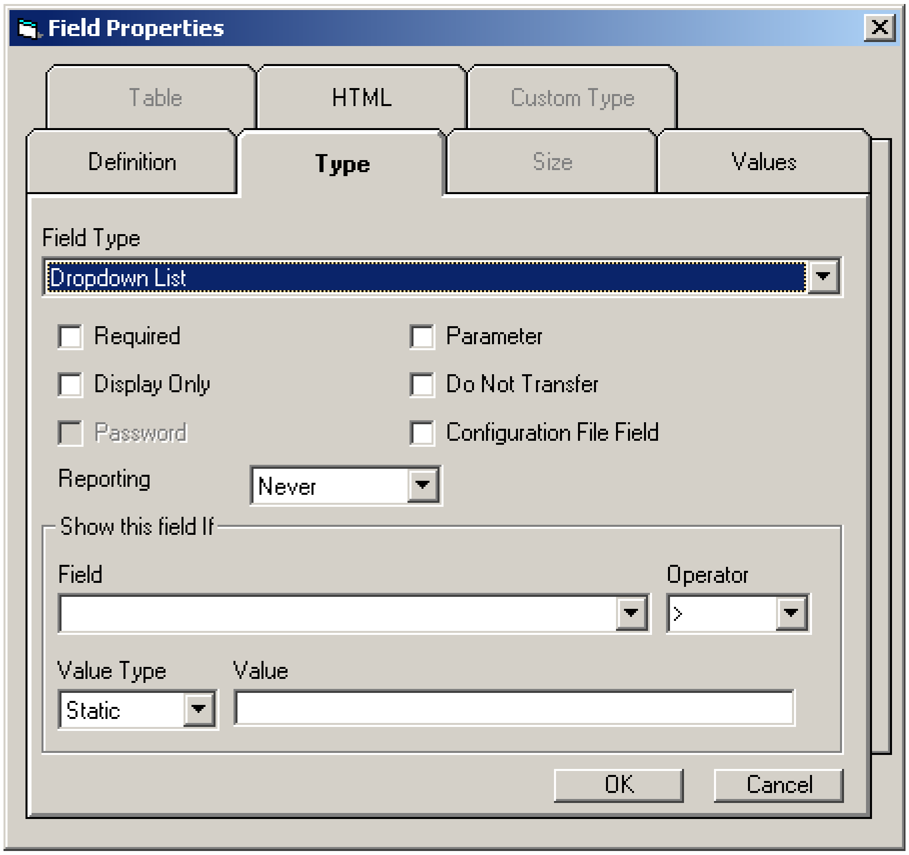
A Dropdown List field will select the customer’s home state (which, for our example, we know to be one of the six New England states).

Procedure:
Create a Dropdown List Field

Start of procedure

1. Click Add on the Field List dialog box to initiate the process for creating a new Field and to open the Field Properties dialog box.
2. In the Field Label field on the Field Properties dialog box, type a label for the Field you are creating.

3. Click the `Field Name` field or `Tab` to it to populate this field with a name that corresponds to the label you just entered.
4. Click the `Type` tab on the `Field Properties` dialog box to select the Field type.



Field Type: Dropdown List

□ Required
□ Parameter
□ Display Only
□ Do Not Transfer
□ Password
□ Configuration File Field

Reporting: Never

Field:

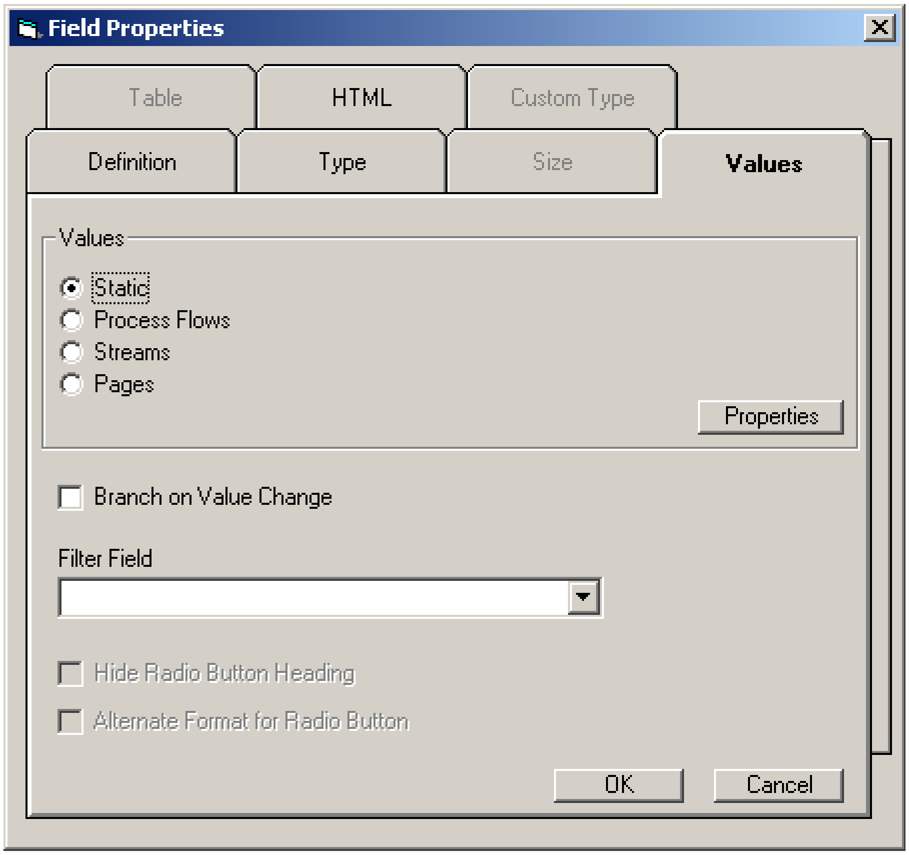
Operator: >

Value Type: Static

Value:

Figure 28: Field Properties Dialog Box - Dropdown List Field Type

5. Select Field Type `Dropdown List` from the dropdown list as shown in Figure 28.
6. Click the `Values` tab on the `Field Properties` dialog box to specify values for this Dropdown List.



Values

● Static
○ Process Flows
○ Streams
○ Pages

□ Branch on Value Change

Filter Field

□ Hide Radio Button Heading

□ Alternate Format for Radio Button

Figure 29: Field Properties Dialog Box - Dropdown List Values

7. In the `Values` area, select `Static`.

   Static informs the application that the value will not change.

8. Click `Properties`, then click `Yes` to save the Field if a message appears.

9. Click `Add` on the `Field Value List` dialog box as shown in Figure 30 to initiate the process for assigning values to this Dropdown List, and to open the `Field Value` dialog box.

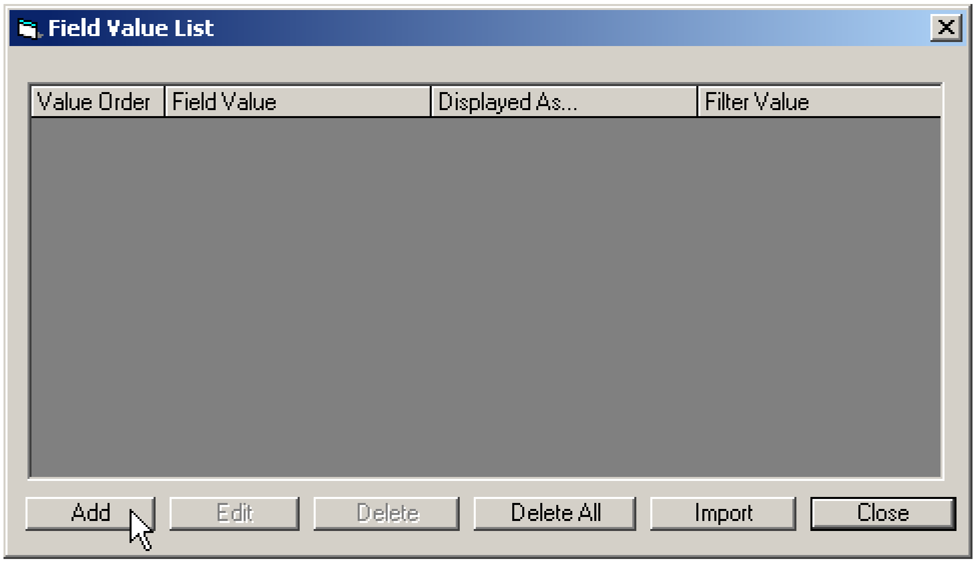
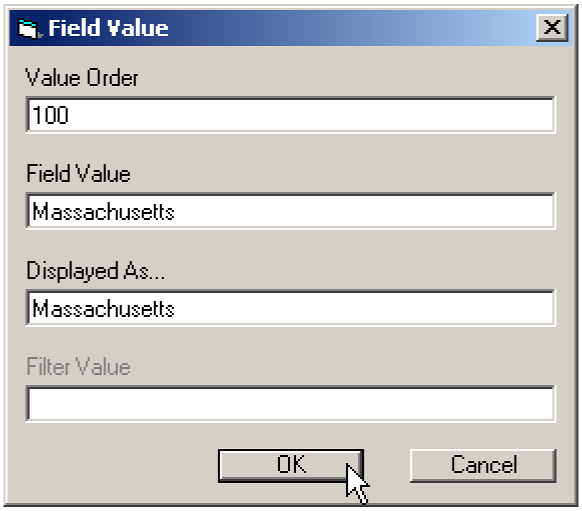


Figure 30: Field Value List Dialog Box

10. In the Field Value field on the Field Value dialog box, type a display value for the Dropdown List option you are creating (such as Massachusetts).
11. Click the Displayed As... field or Tab to it to populate this field with a name that corresponds to the display value you just entered, as shown in Figure 31.



Value Order: 100

Field Value: Massachusetts

Displayed As...: Massachusetts

Filter Value:

Figure 31: Field Value, Massachusetts

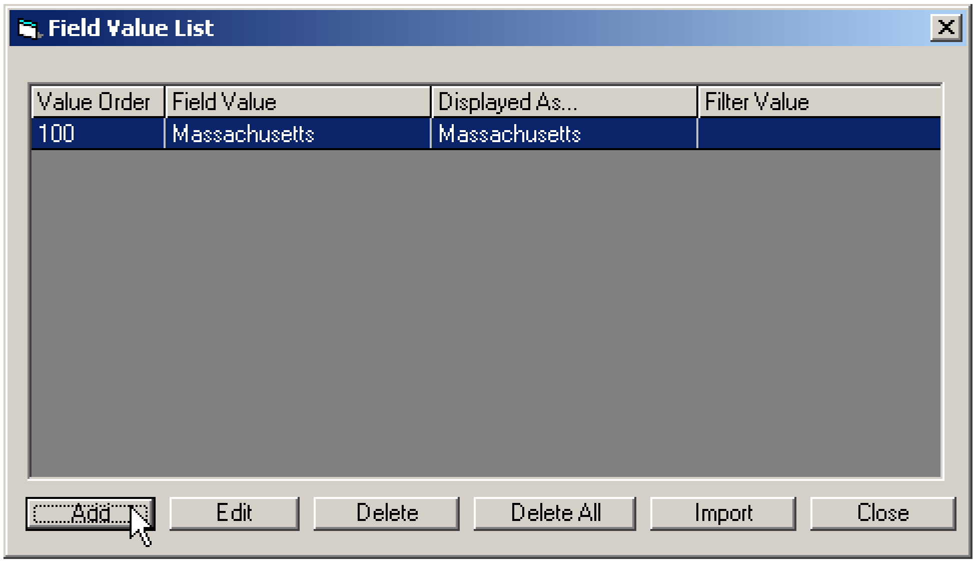
The Value Order allows you to order the items in the Dropdown List (smallest to largest).

The Field Value holds the name that will be displayed for the option on the web page.

The `Displayed As...` defines how the selection is referenced from within Agent Scripting.

12. Click `OK`.

    Your new Dropdown List value is added to the `Field Value List` as shown in Figure 32.

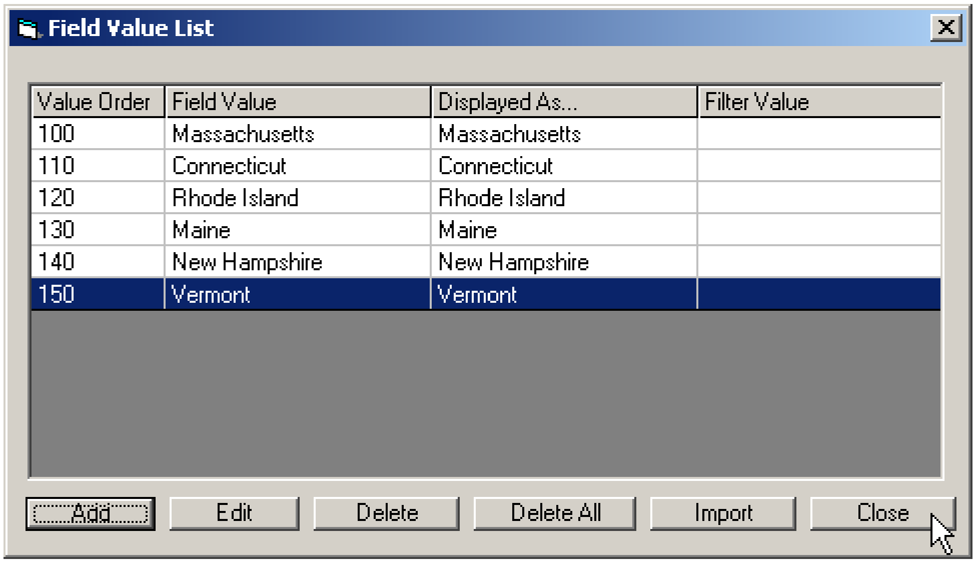


Value Order	Field Value	Displayed As...	Filter Value
100	Massachusetts	Massachusetts	

Figure 32: Field Value List

13. You need to add five additional values for this Dropdown List. Repeat Steps 9 through 12 using the following `Field Value` and `Displayed As...`:
    - `Connecticut`
    - `Rhode Island`
    - `Maine`
    - `New Hampshire`
    - `Vermont`

    The `Field Value List` appears as shown in Figure 33.



Value Order	Field Value	Displayed As...	Filter Value
100	Massachusetts	Massachusetts	
110	Connecticut	Connecticut	
120	Rhode Island	Rhode Island	
130	Maine	Maine	
140	New Hampshire	New Hampshire	
150	Vermont	Vermont	

Figure 33: Field Value List - Dropdown List Values Added

14. Click `Close` on the `Field Value List` after all six values are added.

15. Click `OK` on the `Field Properties` dialog box to complete your Dropdown List.

End of procedure

Your new Field is added to the `Field List`.

Chapter

5 Creating Actions

This chapter provides an overview of the process of creating Actions, setting Action properties, and using the Action wizard.

The information in this chapter is divided among the following topics:

What Is an Action?

An Action is a function that is performed when a button is clicked or when a Page is loaded or unloaded.

Types of Actions

There are many types of Actions that you can create and use within the Genesys Agent Scripting Development Environment. The Action wizard can help you create many types of Actions, including:

- Add Row
- API Interface
- Branch
- Database Interface
- Delete Row
- Insert Row
- Load Session

- Next
- Previous
- Save Session
- Unselect All Rows
- Update Row
- XML Interface

Creating an Action

Procedure:
Create an Action

Purpose: To create a new Action, using the `Define` menu.

Start of procedure

1. Select `Define` > `Actions` to open the `Action List` dialog box as shown in Figure 34.

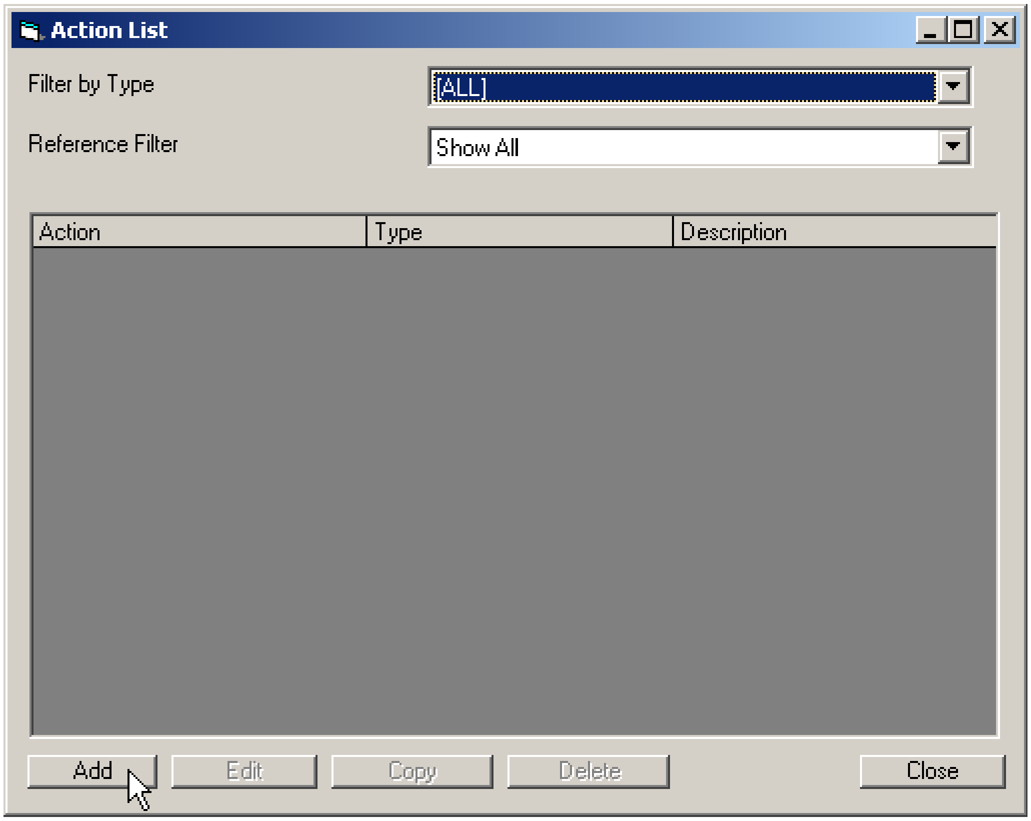
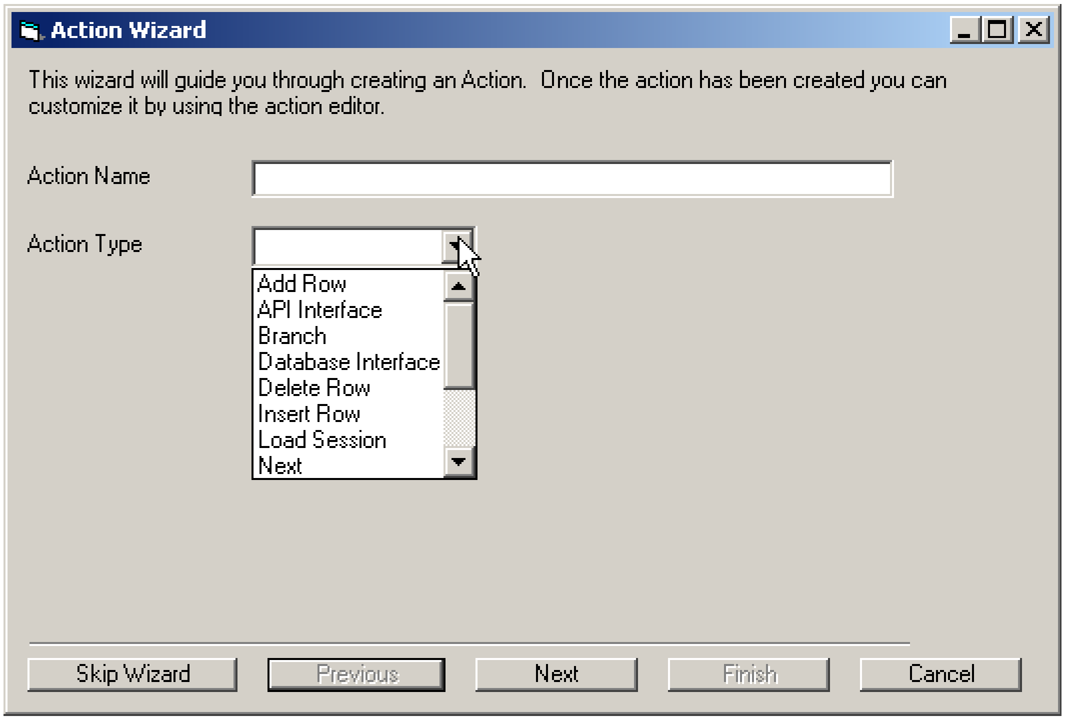


Figure 34: Action List Dialog Box

2. Click `Add` to initiate the process for creating a new Action.

   The `Action Wizard` dialog box opens as shown in Figure 35.



This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

Figure 35: Action Wizard Dialog Box

End of procedure

Next Steps

The Action wizard allows you to set up many Action types in a step-by-step fashion.

Action Wizard

Table 24 describes the options on the initial screen of the Action wizard.

Table 24: Action Wizard - Initial Screen

Control	Description / What to Do...
Action Name	A text field that holds the name of this Action, which displays as the name of the button. This is the name of the Action that appears in and is referenced by the Agent Scripting Editor. Type a name for this Action.
Action Type	Select the Action type for this Action from the dropdown list. See "Descriptions of Action Types" on page 112 for more information on Action types.
Skip Wizard	If you want more manual control over the creation and definition of any other Action type, click Skip Wizard to open the Action Properties dialog box.
Previous	As you define the specific details of an Action using the Action wizard, click the Previous button to review or make changes for the previous step.
Next	If the Action type you are defining requires you to proceed to additional Action wizard screens, click the Next button to go to the next step.
Finish	The Finish button becomes available when the Action wizard has enough information to define the Action type you selected. Click Finish to complete the Action.

Descriptions of Action Types

Add Row

The Add Row action adds a row to a table. The new row will be populated with the current values of the Fields that are contained in the table. See "Table Actions" on page 117.

API Interface

The API Interface action calls and executes a previously-defined API Interface. See "Interface Actions" on page 119.

Branch

The `Branch` action allows the script user to move to a specified Page. A `Branch` *action* differs from `Page Branch` in that it is only activated when the `Branch` action button is clicked.

As an example, if a customer hangs up in the middle of a call, you can provide an `End Call` button which branches the script user directly to the final Page of the interaction, skipping the usual intervening Pages. See "Navigation Actions" on page 114.

Database Interface

The `Database Interface` action calls and executes a previously-defined Database Interface. See "Interface Actions" on page 119.

Delete Row

The `Delete Row` action deletes the current row from a table. See "Table Actions" on page 117.

Insert Row

The `Insert Row` action inserts a new row into a table. The new row is blank and any values entered in the Fields that make up the table but are not yet saved into the table are cleared. See "Table Actions" on page 117.

Load Session

The `Load Session` action loads a complete session. The action uses the value within a field to uniquely identify the session. See "Session Actions" on page 123.

Next

The `Next` action allows the script user to move from one Page to the next logical page in the interaction (either the next physical Page in the Stream, or to the Page specified in a branching condition).

Use the `Next` action when you want to transition from one Page to the next. See "Navigation Actions" on page 114.

Previous

The `Previous` action allows the script user to go back to the previous Page. See "Navigation Actions" on page 114.

Save Session

The `Save Session` action saves a complete session. The action uses the value within a field to uniquely identify the session. See "Session Actions" on page 123.

Unselect All Rows

The `Unselect All Rows` action clears all rows in a table.

`Unselect All Rows` causes the same behavior as `Insert Row` for the component Fields that make up a table, and unselects all of the rows in the table. The corresponding component Field values are reset. See "Table Actions" on page 117.

Update Row

The `Update Row` action updates the current row in a table with values from Fields contained in the table.

`Update Row` causes the same behavior as `Add Row` except that it operates on the currently selected row. See "Table Actions" on page 117.

XML Interface

The `XML Interface` action calls and executes a previously-defined XML Interface. See "Interface Actions" on page 119.

Navigation Actions

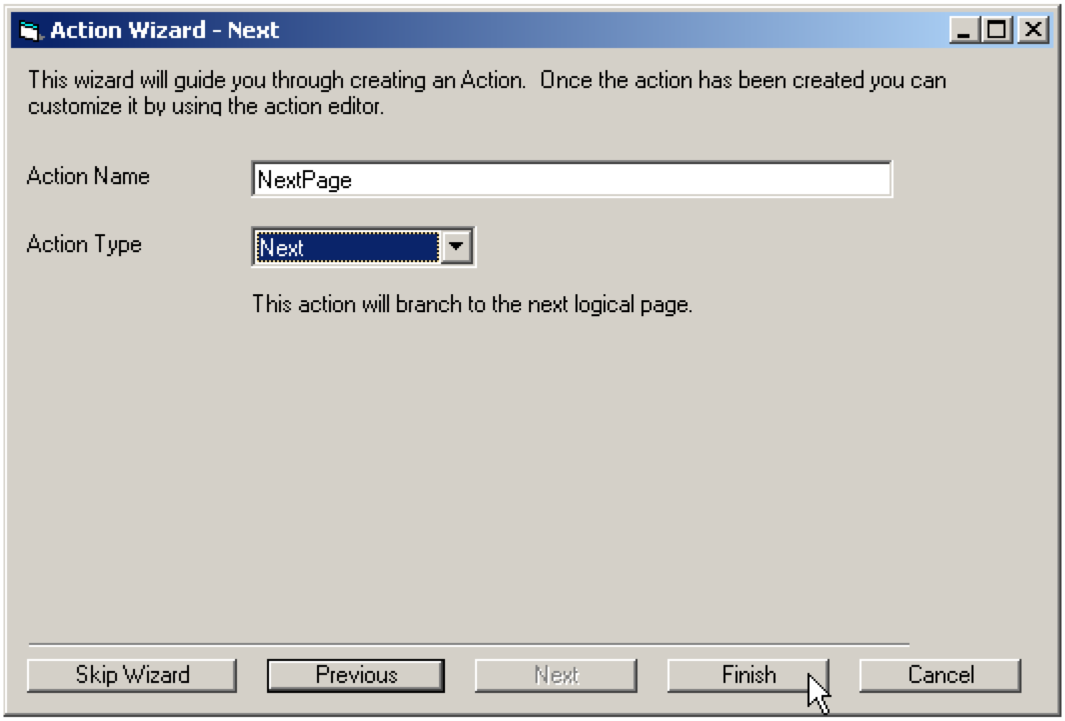
Navigation Actions include:

- `Previous`
- `Next`
- `Branch`

Previous and Next

`Previous` and `Next` actions are straightforward to define and require no additional input in the Action wizard.

If you select `Previous` or `Next` in the `Action Type` field, click `Finish` to complete the Action as shown in Figure 36.



This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

Action Name: NextPage

Action Type: Next

This action will branch to the next logical page.

Figure 36: Action Wizard - Next

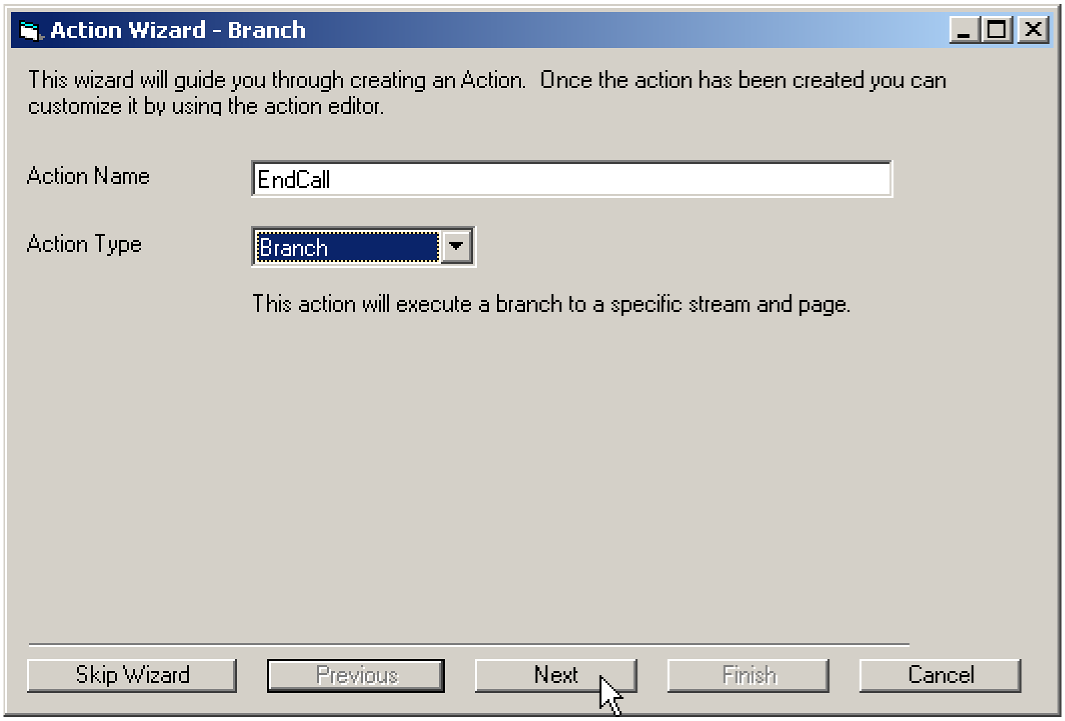
If you want to customize or modify the properties of the Action, select the `Edit Action` check box and click `Close` to open the `Action Properties` dialog box.

If you are finished defining the Action, unselect the `Edit Action` check box, then click `Close`.

Branch

Branch actions need only specify the Process Flow, Stream, and individual Page to branch to.

If you select Branch in the Action Type field, click Next to continue defining the Action as shown in Figure 37.



Action Wizard - Branch

This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

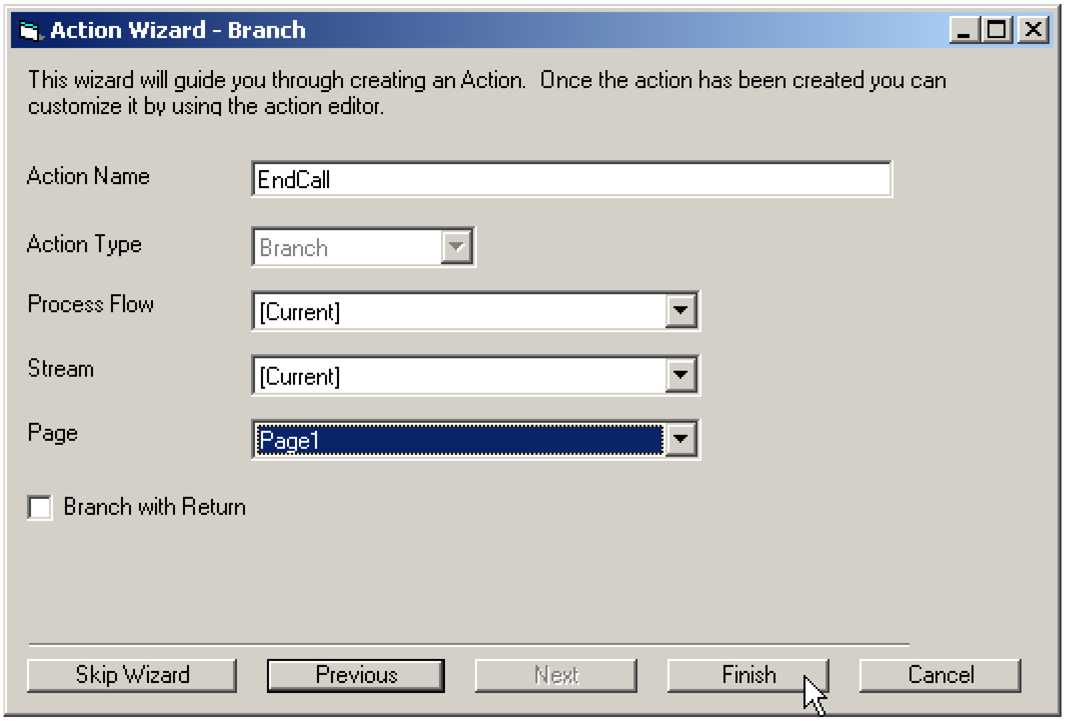
Action Name: EndCall

Action Type: Branch

This action will execute a branch to a specific stream and page.

Figure 37: Action Wizard - Branch

Identify the Process Flow and Stream, then select the individual Page to branch to and click Finish as shown in Figure 38 on page 117.



Action Wizard - Branch

This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

Action Name: EndCall

Action Type: Branch

Process Flow: [Current]

Stream: [Current]

Page: Page1

□ Branch with Return

Figure 38: Action Wizard - Branch, Identify Branch Location

If you want to customize or modify the properties of the Action, select the `Edit Action` check box and click `Close` to open the `Action Properties` dialog box.

If you are finished defining the Action, unselect the `Edit Action` check box, then click `Close`.

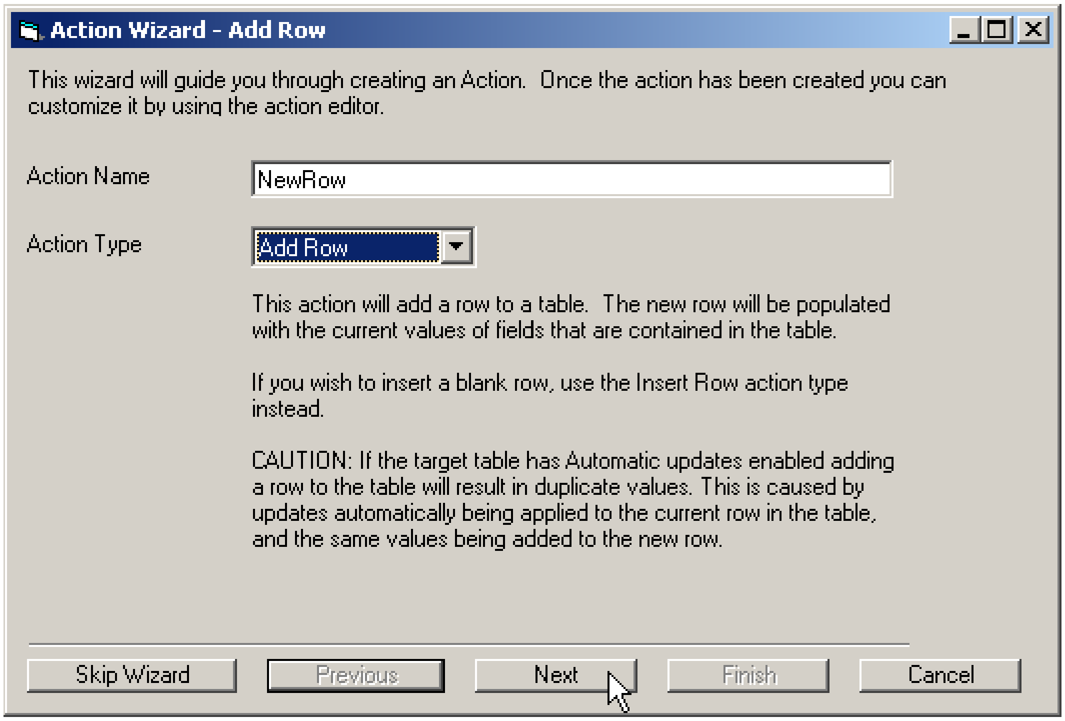
For more information about branching, see the chapter called "Adding Branching Logic" beginning on page 257.

Table Actions

Actions related to a Field of type `Table` include:

- `Add Row`
- `Delete Row`
- `Insert Row`
- Unselect All Rows
- Update Row

If you select any of the above Action types in the `Action Type` field, click `Next` to continue defining the Action as shown in Figure 39 on page 118.



Action Wizard - Add Row

This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

Action Name: NewRow

Action Type: Add Row

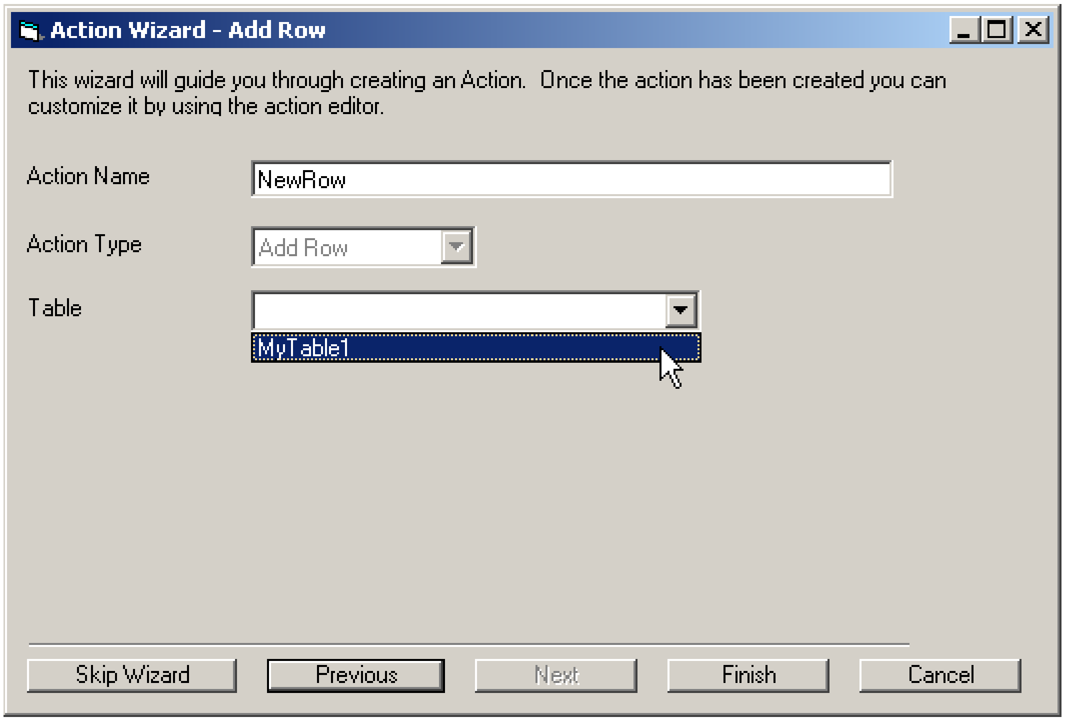
This action will add a row to a table. The new row will be populated with the current values of fields that are contained in the table.

If you wish to insert a blank row, use the Insert Row action type instead.

CAUTION: If the target table has Automatic updates enabled adding a row to the table will result in duplicate values. This is caused by updates automatically being applied to the current row in the table, and the same values being added to the new row.

Figure 39: Action Wizard - Add Row (Example)

Select the `Table` field from the `Table` dropdown list and click `Finish` as shown in Figure 40.



Action Wizard - Add Row

This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

Action Name: NewRow

Action Type: Add Row

Table: MyTable1

Figure 40: Action Wizard - Add Row, Select Table

If you want to customize or modify the properties of the Action, select the `Edit Action` check box and click `Close` to open the `Action Properties` dialog box.

If you are finished defining the Action, unselect the `Edit Action` check box, then click `Close`.

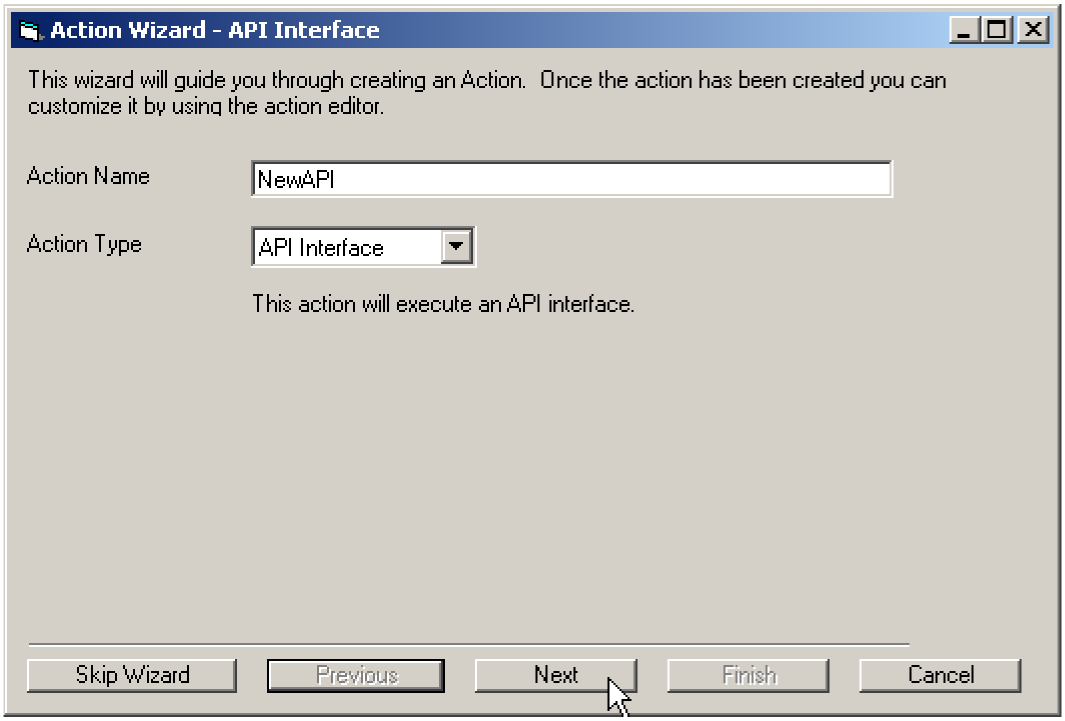
Interface Actions

Interface actions include:

- `API Interface`
- `Database Interface`
- `XML Interface`

API Interface

If you select the `API Interface` Action type in the `Action Type` field, click `Next` to continue defining the Action as shown in Figure 43.



This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

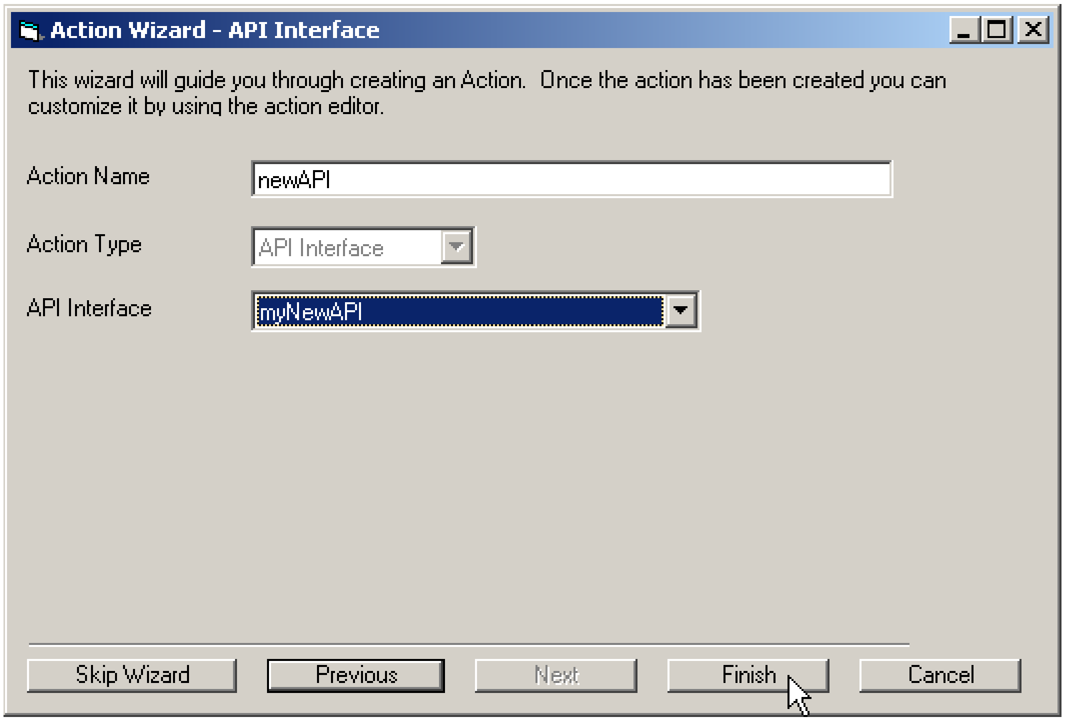
Action Name: NewAPI

Action Type: API Interface

This action will execute an API interface.

Figure 41: Action Wizard - API Interface (Example)

Select the API Interface from the `API Interface` dropdown list and click `Finish` as shown in Figure 44 on page 121.



This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

Action Name: newAPI

Action Type: API Interface

API Interface: myNewAPI

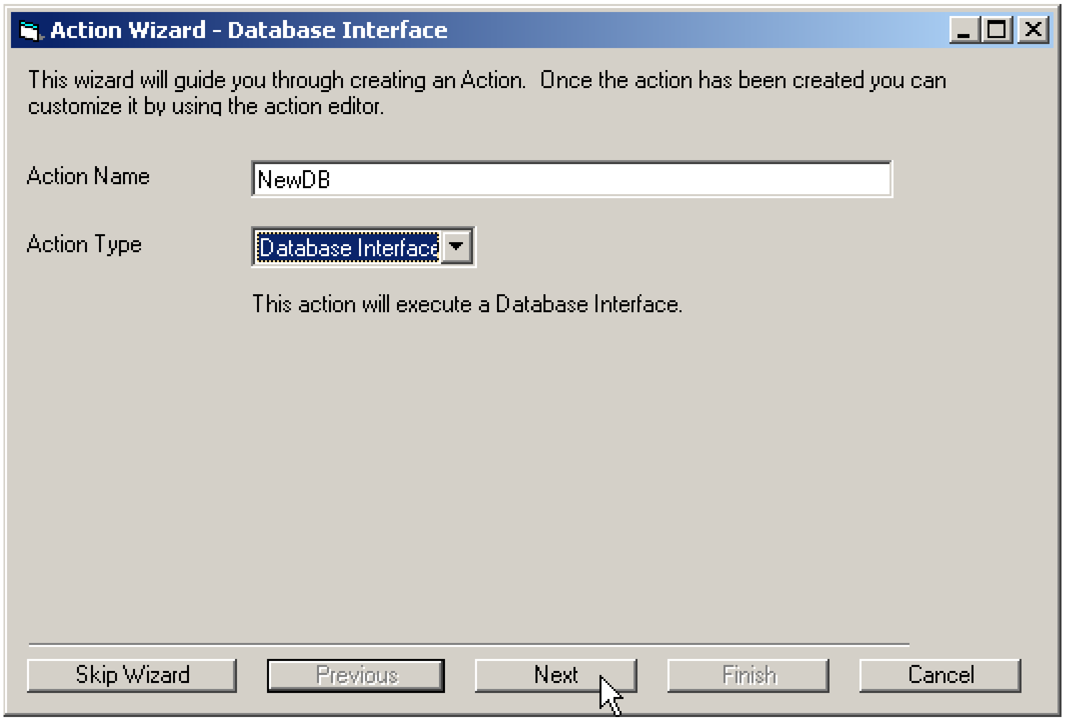
Figure 42: Action Wizard - API Interface, Select API Interface

If you want to customize or modify the properties of the Action, select the `Edit Action` check box and click `Close` to open the `Action Properties` dialog box.

If you are finished defining the Action, unselect the `Edit Action` check box, then click `Close`.

Database Interface

If you select the `Database Interface` Action type in the `Action Type` field, click `Next` to continue defining the Action as shown in Figure 43.



This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

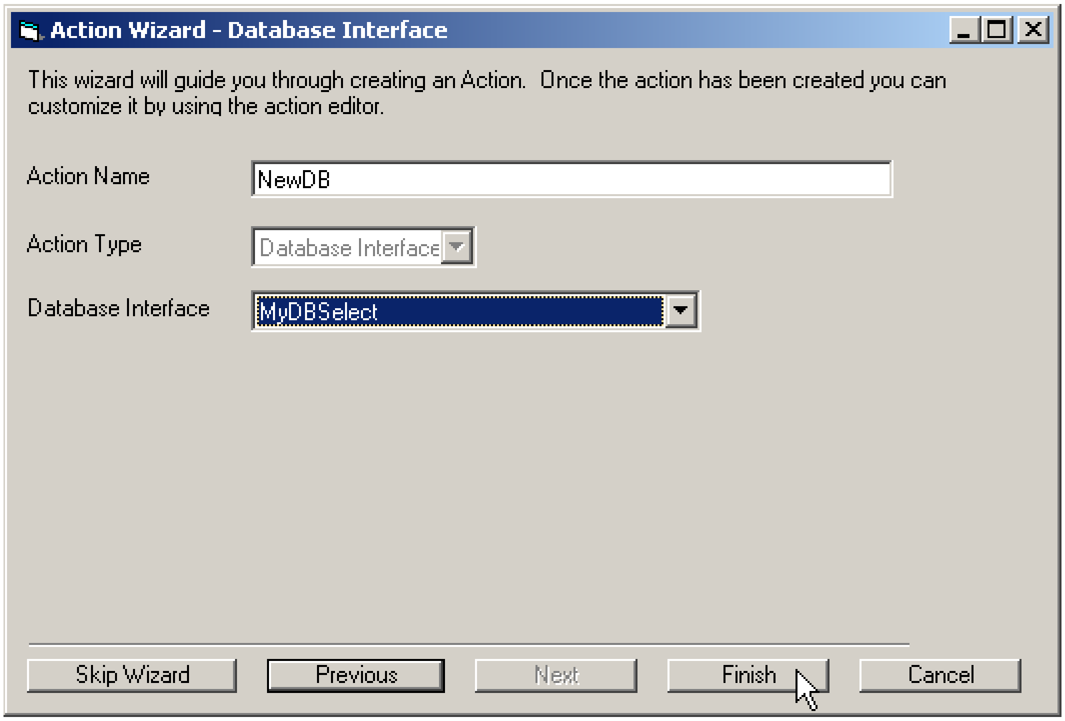
Action Name NewDB

Action Type Database Interface

This action will execute a Database Interface.

Figure 43: Action Wizard - Database Interface (Example)

Select the Database Interface from the `Database Interface` dropdown list and click `Finish` as shown in Figure 44.



This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

Action Name NewDB

Action Type Database Interface

Database Interface MyDBSelect

Figure 44: Action Wizard - Database Interface, Select Database Interface

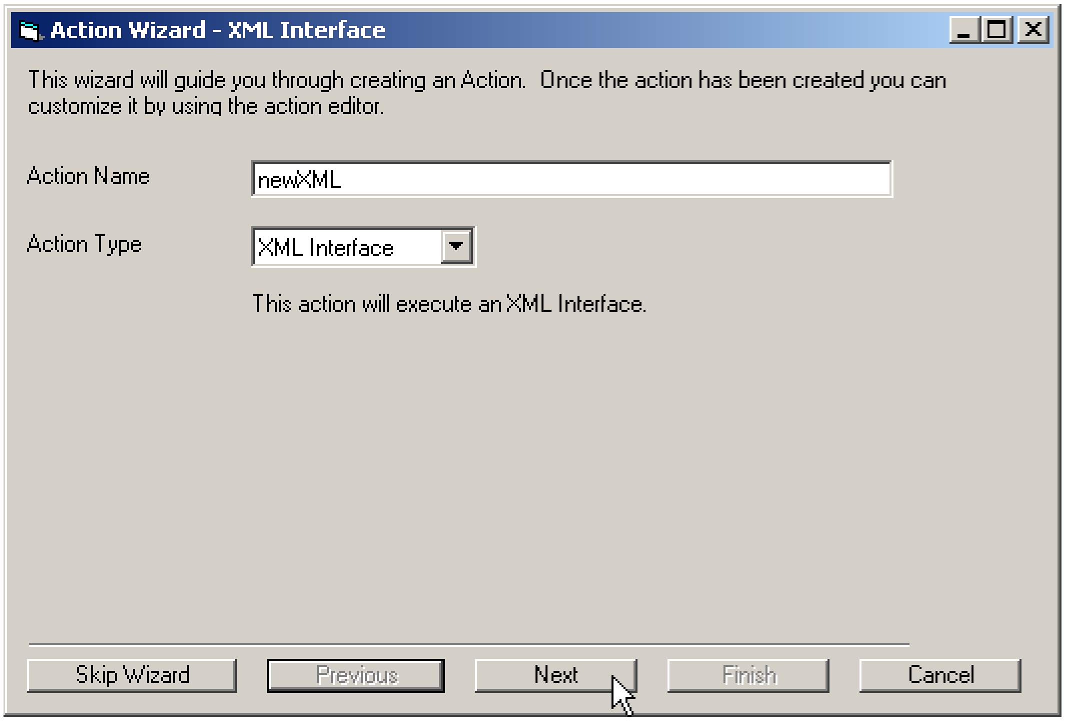
If you want to customize or modify the properties of the Action, select the Edit Action check box and click Close to open the Action Properties dialog box.

If you are finished defining the Action, unselect the Edit Action check box, then click Close.

See the chapter called "Databases and Database Interfaces." beginning on page 171 for more details.

XML Interface

If you select the XML Interface Action type in the Action Type field, click Next to continue defining the Action as shown in Figure 43.



Action Wizard - XML Interface

This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

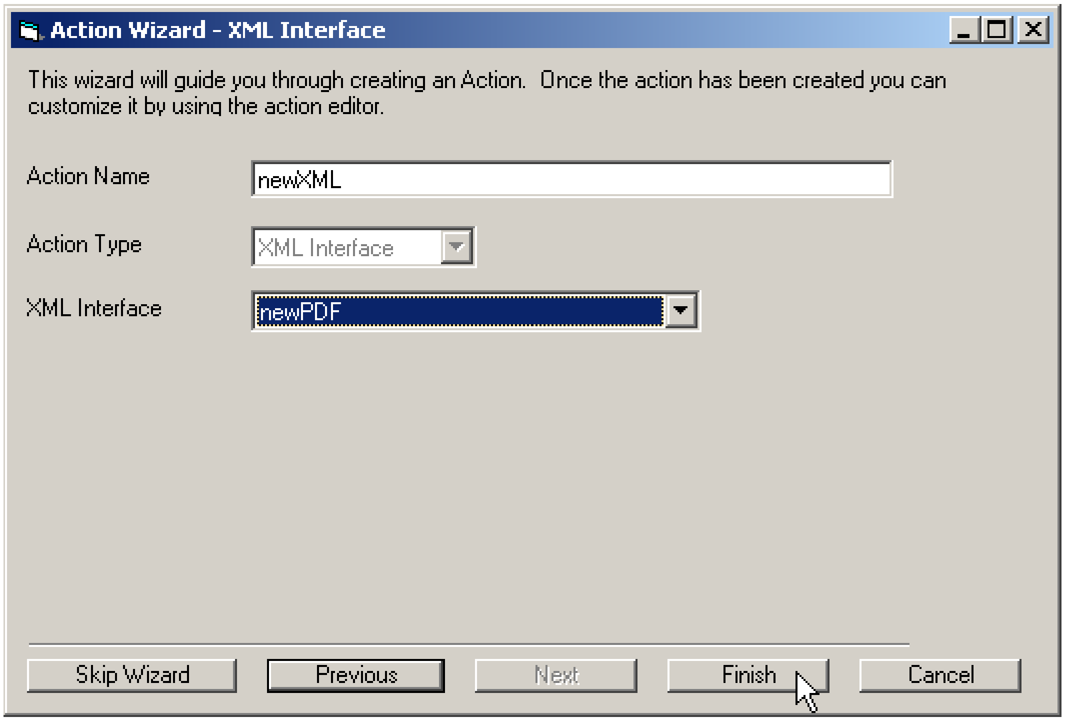
Action Name: newXML

Action Type: XML Interface

This action will execute an XML Interface.

Figure 45: Action Wizard - XML Interface (Example)

Select the XML Interface type you need from the XML Interface dropdown list and click Finish as shown in Figure 44 on page 121.



Action Wizard - XML Interface

This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

Action Name: newXML

Action Type: XML Interface

XML Interface: newPDF

Figure 46: Action Wizard - XML Interface, Select XML Interface Type

If you want to customize or modify the properties of the Action, select the `Edit Action` check box and click `Close` to open the `Action Properties` dialog box.

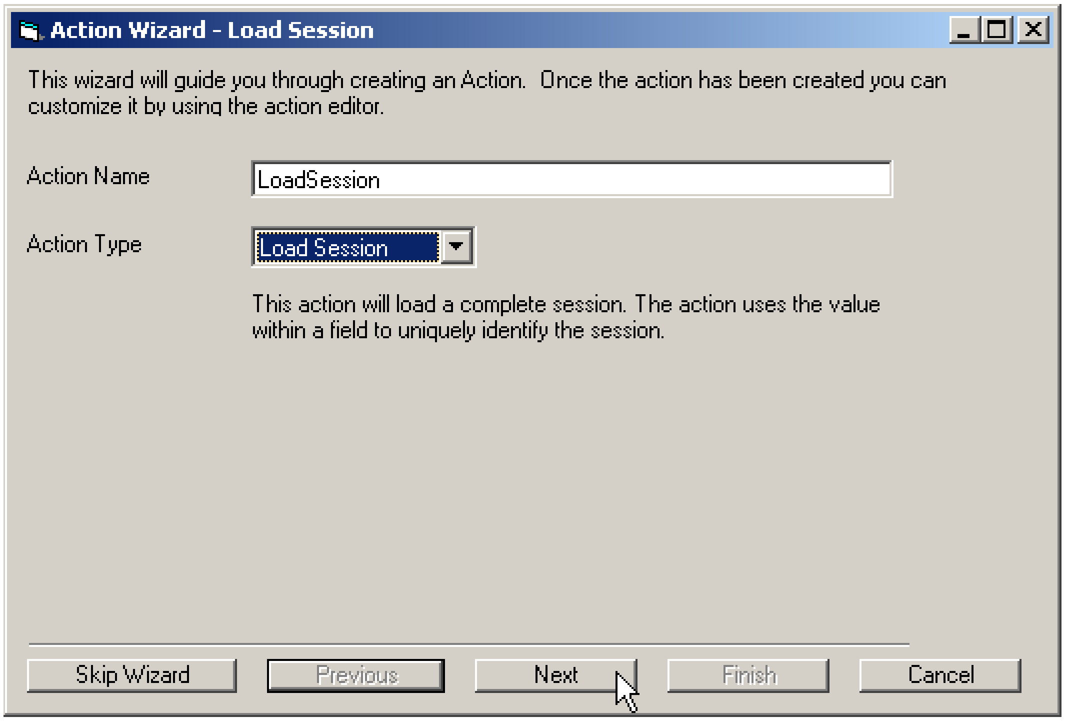
If you are finished defining the Action, unselect the `Edit Action` check box, then click `Close`.

Session Actions

Session actions include:

- `Load Session`
- `Save Session`

If you select either of the above action types in the `Action Type` field, click `Next` to continue defining the Action as shown in Figure 47 on page 124.



Action Wizard - Load Session

This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

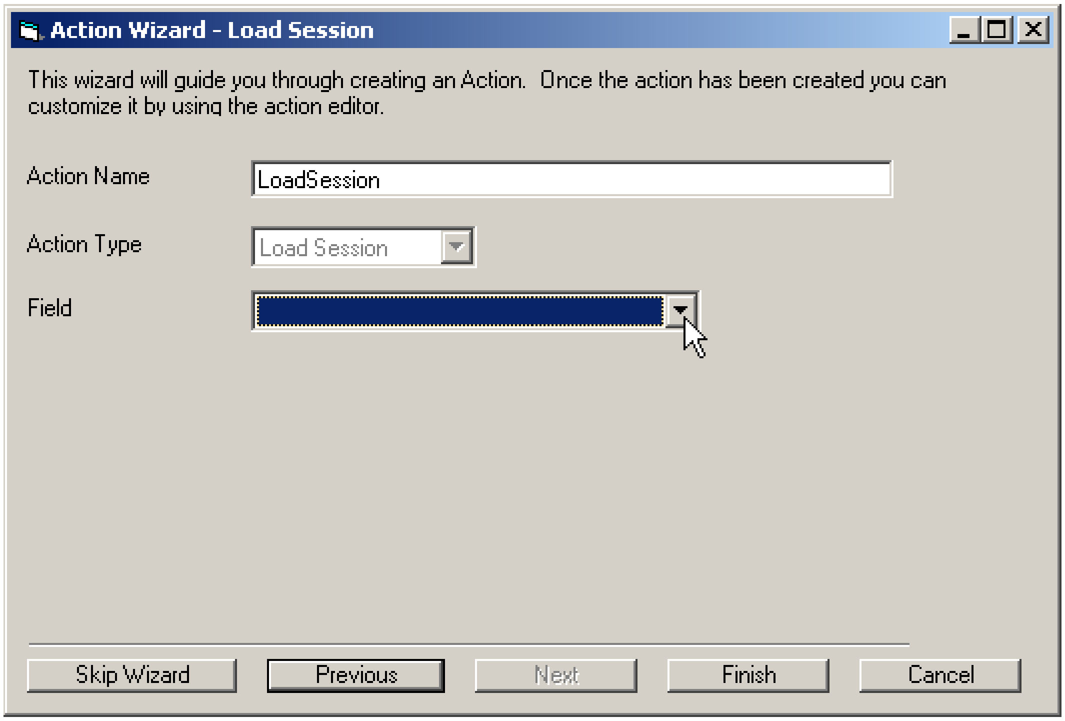
Action Name: LoadSession

Action Type: Load Session

This action will load a complete session. The action uses the value within a field to uniquely identify the session.

Figure 47: Action Wizard - Load Session (Example)

Select the Field containing the value for the session from the `Field` dropdown list and click `Finish` as shown in Figure 48.



Action Wizard - Load Session

This wizard will guide you through creating an Action. Once the action has been created you can customize it by using the action editor.

Action Name: LoadSession

Action Type: Load Session

Field:

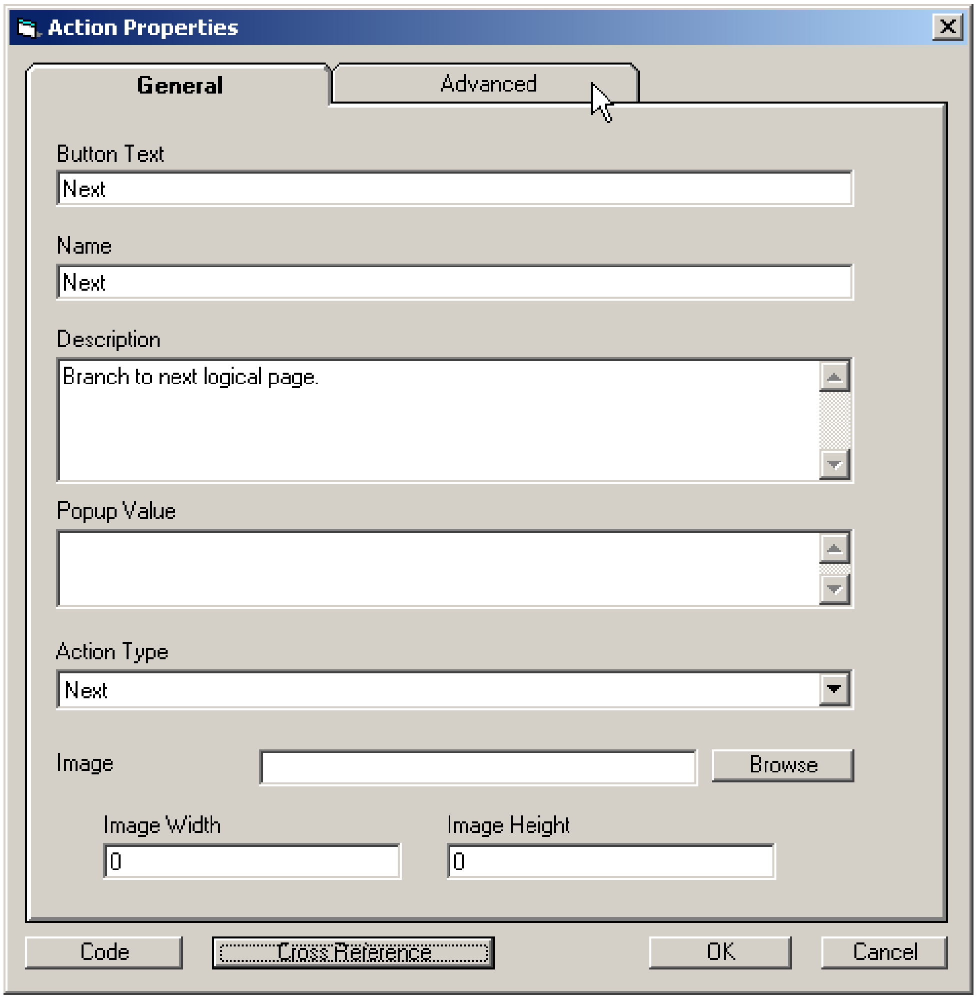
Figure 48: Action Wizard - Load Session, Select Field

If you want to customize or modify the properties of the Action, select the Edit Action check box and click Close to open the Action Properties dialog box.

If you are finished defining the Action, unselect the Edit Action check box, then click Close.

Action Properties

The Action Properties dialog box is shown in Figure 49.



Button Text: Next

Name: Next

Description: Branch to next logical page.

Popup Value:

Action Type: Next

Image:

Image Width: 0

Image Height: 0

Figure 49: Action Properties Dialog Box

The Action Properties dialog box is where you define the characteristics that are unique to each Action you wish to create. It opens if you click Skip Wizard at the initial screen of the Action Wizard dialog box, and also if you leave the Edit Action check box selected when you complete setup of an Action in the Action wizard.

Action Properties General Tab

Table 25 on page 126 describes the options on the General tab of the Action Properties dialog box.

Table 25: Action Properties, General Tab

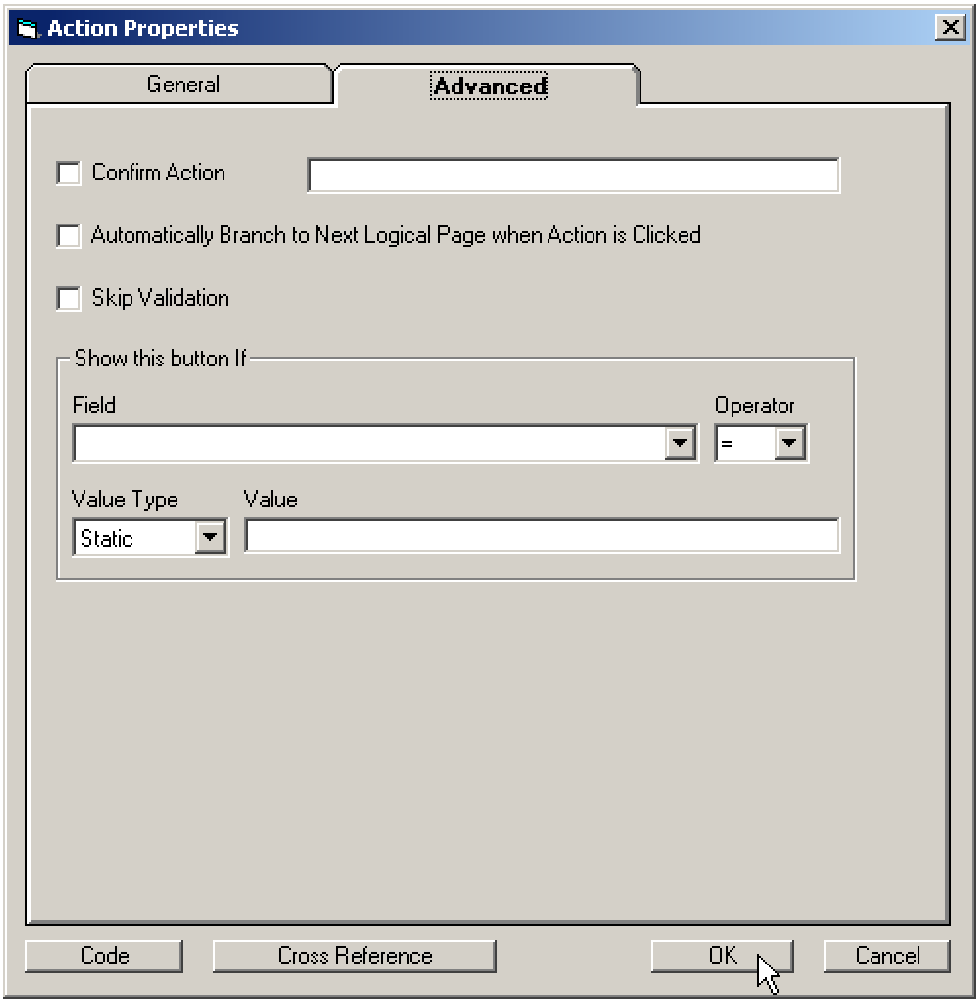
Control	Description / What to Do...
Button Text	This value is displayed as the text for the button, unless a button image is provided.
Name	Use this field to specify a name for the Action. This name appears in the Action List and is used internally. Action names cannot contain spaces and must be unique.
Description	This field contains a description of the Action. It is optional and only used for informational purposes. If the Action is created through the Action wizard, a description is automatically provided.
Popup Value	You can specify a popup value for actions. When you move the mouse over this button at runtime, a popup will appear with this value.
Action Type	The Action Type indicates how the Action is used. Valid Action types are described in “Descriptions of Action Types” on page 112. Action types are also used to organize and filter Actions in the Action List. You can use the Action wizard to define most Action types.
Image	Actions can be displayed as images rather than standard buttons by providing the name of an image file.
Browse	Click Browse to navigate to a desired image file.
Image Width	If the Action is being displayed as an image, the Image Width field is used to set the width of the Action image (in pixels).
Image Height	If the Action is being displayed as an image, the Image Height field is used to set the height of the Action image (in pixels).

Table 25: Action Properties, General Tab (Continued)

Control	Description / What to Do...
Code	Click `Code` to open the `Code for Action <action name>` dialog box as shown in Figure 51. This dialog box allows you to select and configure commands that further define your Action.
Cross Reference	The `Cross Reference` button allows you to see where an Action is being used.
OK	Click `OK` when you are ready to complete setup of your new Action.

Action Properties Advanced Tab

The Advanced tab of the Action Properties dialog box is shown in Figure 49.



□ Confirm Action

□ Automatically Branch to Next Logical Page when Action is Clicked

□ Skip Validation

Show this button If

Field

Operator: =

Value Type: Static

Value

Figure 50: Action Properties Dialog Box - Advanced Tab

Table 26 on page 129 describes the options on the Advanced tab of the Action Properties dialog box.

Table 26: Action Properties, Advanced Tab

Control	Description / What to Do...
Confirm Action	Select the `Confirm Action` check box to specify that, at runtime, a script user must confirm the Action. If this check box is selected, the adjacent text field becomes active and allows you to specify what message appears in the runtime confirmation dialog box. The default message is: `Are you sure?` and the script user must click `Yes` to proceed.
Automatically Branch to Next Logical Page when Action is Clicked	Select this check box to cause Genesys Agent Scripting to generate a branch condition when the Action button is clicked. When a branch condition is generated, then any branch conditions on the Page containing the Action are first checked. If no branch conditions are satisfied, then the branch is to the next logical page as defined in the `Project Tree` and `Page Tree.`
Skip Validation	Select this check box to cause Genesys Agent Scripting to skip all validation at runtime. This allows users to leave a Page that has required fields without providing the values. This works well for `Cancel` buttons. **Note:** If this check box is selected and the Action is called in a `Page Unload` command, the validation will *not* be skipped.
Show This Button If	Buttons can be shown or hidden based on the value of a specific Field. See “Making an Action Context-Sensitive” on page 140 for more information.
Field	This is the field whose value will determine whether the button you are currently defining will be shown or hidden. Select a Field from the `Field` dropdown list.

Table 26: Action Properties, Advanced Tab (Continued)

Control	Description / What to Do...
Operator	Depending on the Field chosen above, Genesys Agent Scripting automatically determines the valid operators and range of values. Select the appropriate operator from the dropdown list here. For example, if you select a `Check Box` Field type, then only the equal (=) and not equal (<>) operators will be selectable, and only `True` and `False` for values will be selectable. If you select a `Radio Button` Field type, only the equal (=) and not equal (<>) operators will be selectable, and the `Value` dropdown list will list all possible values for the Radio Button.
Value Type	Select either a `Field` or a `Static` value type from the dropdown list to compare your Field against.
Value	Type or select the appropriate value in this field.

Code for Action

Figure 51 shows the `Code for Action <action name>` dialog box. A `Function` action is the Action type that you are most likely to define through the `Code` button on `Action Properties`.

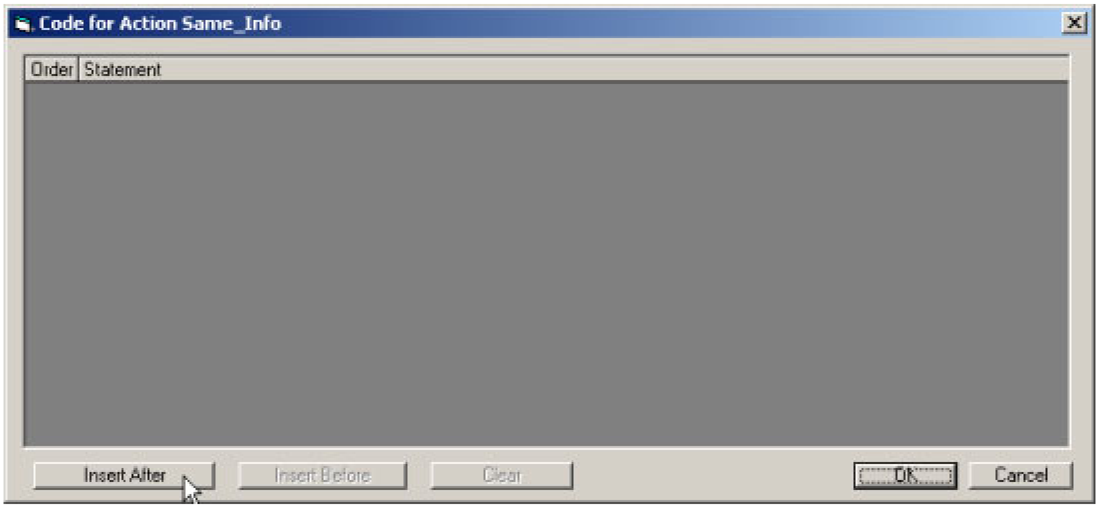
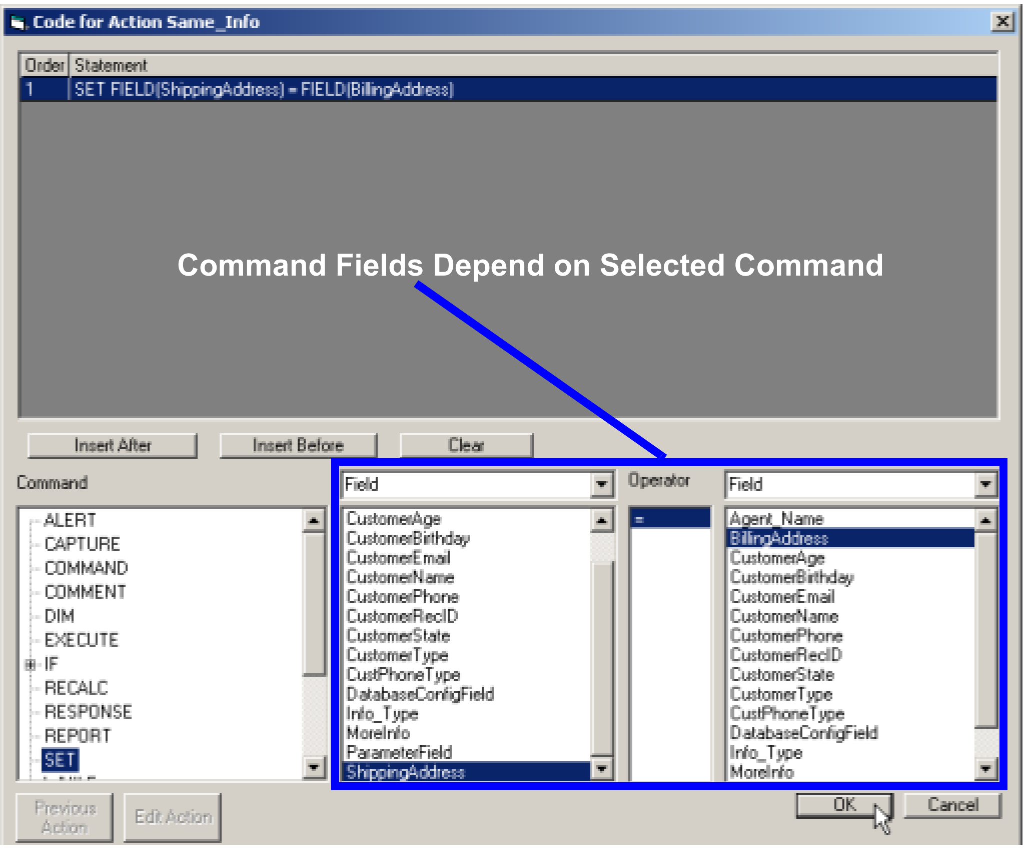


Figure 51: Code for Action Dialog Box

Click `Insert After` to open the dialog box shown in Figure 52 on page 131, where you set up commands and parameters for your Action.



Code for Action Same_Info

Order	Statement
1	SET FIELD(ShippingAddress) = FIELD(BillingAddress)

Command Fields Depend on Selected Command

Figure 52: Code for Action Dialog Box - Insert a Command

Table 27 on page 132 describes the options and values on the `Code for Action <action name>` dialog box.

Table 27: Code for Action

Control	Description / What to Do...
Order/Statement	This is the series of commands that define this Action and the order they are executed. Some actions may require that you define a sequence of several commands.
Insert After	Click `Insert After` for the next command to be inserted after the command that is currently selected in the display field under `Order` and `Statement`.
Insert Before	Click `Insert Before` for the next command to be inserted before the command that is currently selected in the display field under `Order` and `Statement`.
Clear	Click `Clear` to delete the command that is currently selected in the display field under `Order` and `Statement`.

Table 27: Code for Action (Continued)

Control	Description / What to Do...
Command	Select the command for this Action from the tree list on the left. The command fields that appear on the right will change depending on the particular command you select. These variable command fields usually consist of a dropdown list and a box from which you select an option or type information (as shown for the SET command in Figure 52 on page 131). Command fields contain several possible elements and have the following meanings: Field = one of the previously-defined data Fields. Select Field from the dropdown list and then select the particular Field in the box. Free Form = enter any value (enclosed by quotation marks). Select Free Form from the dropdown list and then type a quoted string. Action = one of the previously-defined Actions. Select Action from the dropdown list and then select the particular Action in the box. Database Interface = one of the previously-defined Database Interfaces. Select Database Interface from the dropdown list and then select the particular Database Interface in the box. Operator = creates a relationship between the two command Fields. Select an Operator to activate it. Page = Selects a Page within the Project that is the target of the command. Select from the Process Flow, Stream, and Page dropdown lists to select the desired Page. Static Value = enables you to select a previously-created static value that is associated with the Field. For example, a Radio Button's Field properties allow assignment of static values. Table = one of the previously-defined Table fields. Select Table from the dropdown list and then select the particular Table field in the box. See "Descriptions of Commands" on page 134 for more information on the available commands.

Table 27: Code for Action (Continued)

Control	Description / What to Do...
Previous Action	`Previous Action` is used when an `Execute` command has been selected and you click `Edit Action` to edit the Action specified in the command. `Previous Action` allows you to return to the previous Action definition.
Edit Action	`Edit Action` is used when an `Execute` command has been selected. It will open up the definition of the Action for editing that is the target of the `Execute` statement.
OK	Click `OK` to complete the command definition.

Descriptions of Commands

Individual Commands

ALERT

Displays a message to the script user in an alert box.

You can cause an alert box to appear and display a Genesys Agent Scripting Field or free form text.

CAPTURE

Captures the content of a Page.

You can capture the content of a Page to a Field or free form text. It can then be used to create an e-mail, PDF, and so on. Capturing a Page also captures any style sheets or images.

COMMAND

Executes static text as a command in the target platform; that is, *VBScript* for ASP, *C#* (commonly referred to as *C-Sharp*) for ASPX, and *Java* for JSP.

This Field allows you to insert additional code in the target language.

COMMENT

This command allows you to insert comments into the code.

DIM

This allows you to manually declare a variable. Variables do not normally need to be declared explicitly. However, declaring a variable manually can specify its type when it is ambiguous.

You can define a variable as a String, Boolean, DateTime, or Number.

EXECUTE

This command allows you to execute any predefined Genesys Agent Scripting Actions.

The area to the right of the command list contains a table with the existing Genesys Agent Scripting actions.

RECALC

This command recalculates the value for the specified Genesys Agent Scripting Function Field selected from the dropdown list.

RESPONSE

This command allows you to write a message to the top of the web browser window. The message is in `Free Form` text and can be text only, HTML, or Javascript.

REPORT

This command allows you to set up a code statement for reporting or reporting results.

SET

This command allows you to set a Genesys Agent Scripting Field or free form text equal to another Genesys Agent Scripting Field, free form text, or static value from a dropdown list or set of radio buttons.

If you are working within a table, using the `FOR EACH ROW` command (see "FOR EACH ROW" on page 138), additional options in the dropdown list are also available. You may also set the value for that row's column or status equal to a Field, variable, column, status, or static value where appropriate.

IF Commands

IF

Executes the code that follows if the conditions indicated are true. Conditions can be the comparison of one Field, variable, or static value to another Field, variable or static value.

In an `IF` command, you can compare a Genesys Agent Scripting Field or free form text to another Genesys Agent Scripting Field, free form text, or a list of static values from a dropdown list or radio buttons. Based on the results, the code entered in the lines after may or may not be executed.

ELSE

This command executes the code that follows if the proceeding `IF` condition is not true.

ELSEIF

This command allows you to have multiple conditional statements. If the preceding `IF` or `ELSEIF` statements were not true, it checks to see if this condition is true and executes the code that follows if it is true.

ENDIF

This command is used to terminate an `IF` condition.

You need to place an `ENDIF` at the end of the `IF` statement.

WHILE Commands

WHILE

This command allows you to execute the code that follows while the conditions indicated are true. Conditions can be the comparison of one Field or variable to another Field, variable, or static value.

ENDWHILE

This command is used to terminate a `WHILE` condition.

You need to place an `ENDWHILE` at the end of the `WHILE` statement.

Async Commands

GET ASYNC CLIENT ID

The `Async Client ID` tells outside servers how to contact the session. This command allows you capture the `Async Client ID` and save it to a field.

The `Async Client ID` can be saved either to a field, or as free-form text entered when creating the action.

PULL VALUE

The `PULL VALUE` command takes the value of the `Listener` field and puts it in a regular Genesys Agent Scripting field. This is similar to having a field on the right side of the `SET` command.

The value taken from the `Listener` field can either be saved to a Genesys Agent Scripting field or as free form text entered when creating the action.

PUSH VALUE

The `PUSH VALUE` command pushes the data into the `Listener` field on the server. This is similar to having a field on the left side of the `SET` command.

SUBSCRIBE TO MESSAGE

Asynchronous interfaces allow you to send messages to everyone who is subscribed to a specific client id or subject. The `SUBSCRIBE TO MESSAGE` action command subscribes the user to the subject specified.

The subject can either be the contents of a particular field or free form text entered when creating the action.

UNSUBSCRIBE FROM MESSAGE

The `UNSUBSCRIBE FROM MESSAGE` action command unsubscribes the user from the subject specified. Once a user has been unsubscribed from a subject, the user won’t receive any messages sent to that subject.

The subject can either be the contents of a particular field or free form text entered when creating the action.

Branch Commands

BRANCH

This command allows the user to transfer control to another script page.

Once the `BRANCH` command has been selected, you can then select the Process Flow, Stream, and Page to which to branch.

You can also branch to a Field if the Field is set to contain a Process Flow, Stream, or Page list. The value that the Field gets set to will determine the Page to which the system is directed.

GET CONTEXT

This command saves the context ID of the Process Flow/Stream/Page to a Field. This Field can then be used as input to a branch command to branch to the Page specified.

GET CURRENT PAGE

This command sets the specified `String` field with the name of the current Page, allowing the current Page name to be displayed on the Page if a Catalog is not used.

GET CURRENT PROCESS FLOW

This command sets the specified `String` field with the name of the current Process Flow, allowing the current Process Flow name to be displayed on the Page if a Catalog is not used.

GET CURRENT STREAM

This command sets the specified `String` field with the name of the current Stream, allowing the current Stream name to be displayed on the Page if a Catalog is not used.

NEXT PAGE

This command causes Genesys Agent Scripting to branch to the next script page. The next script page will either be the next Page in the script, or a Page referenced in a branching condition on the current Page.

PREVIOUS PAGE

This command causes Genesys Agent Scripting to branch to the previous script page that was displayed.

RESTART

This command allows you to start a new session from the selected Page. The `RESTART` Action reinitializes all Fields, clears the Page navigation stacks, and starts runtime processing at the defined Page.

Interface Commands

API INTERFACE

This command is used to execute custom code within Genesys Agent Scripting. When this command is selected the screen will be changed to provide a list of existing API Interfaces.

DATABASE INTERFACE

This command is used to run a Database Interface script. When this command is selected, the screen will be changed to provide a list of existing Database Interfaces.

XML INTERFACE

This command is used to run an XML Interface script. When this command is selected the screen will be changed to provide a list of existing XML Interfaces.

Session Commands

LOAD SESSION

This command will load a saved session based on the value of the field selected or text entered below.

This command is used in conjunction with the `Save` command and requires the same field to be selected or text entered in order to load the corresponding session.

SAVE SESSION

This command allows you to save a session. Saving a session will store all information (for example, call history, data entered, script navigation) in a temporary location. The information can be retrieved with the `Load` command.

You should use the field list to associate a field value with the save option. This field value will be used to determine how the information is stored and retrieved with the load function. The value of this field needs to be able to uniquely identify the saved session. You can also use free form text to assign an identifier to uniquely identify the session to be saved.

CLOSE SESSION

This command will close the current browser window and remove the session.

Table Commands

ADD ROW

This command will cause a new row to be added to the selected table. Values for the new row will come from the current values in the Fields associated with the columns in the table.

The `Field` dropdown list allows you to select a table to which to add the row.

DELETE ROW

This command will cause the currently selected row in a table to be deleted.

FOR EACH ROW

This command is used to iterate through each row in a table. If there are nested tables then this command can also be nested to iterate through sub-tables. While you iterate through the rows of a Genesys Agent Scripting table within an Action, you can access columns in the table using column or Field references. If Field reference is chosen, all valid Fields within Genesys Agent

Scripting will be displayed for selection. If column reference is chosen, only the columns in the table will be displayed for selection.

EXIT FOR

This command allows you to exit the FOR EACH ROW command before reaching the last row in the table.

NEXT ROW

This command allows you to advance to the next row in a table.

You need to place a NEXT ROW at the end of the FOR EACH ROW statement.

GET TABLE ROW

This command will return the number of the current table row which is selected. The return value can be placed in a String or Numeric field.

GET TABLE SIZE

This command will return the number of rows in the table. The return value can be placed in a String or Numeric field.

INSERT ROW

This command will cause a new row to be added to its table. Values for the row will be set to blanks for String fields and zero for Numeric fields. Any values entered in the Fields that make up the table but are not yet saved into the table will be cleared.

If you wish to insert a row with the current values of Fields that are contained in the tables, use ADD ROW instead.

REMOVE ALL ROWS

This command allows you to remove all rows from the specified table including any rows from child tables.

SET TABLE ROW

This command will take a number from a String or Numeric field as input and set the current row in the table to the supplied value. It will also populate the associated Fields in the table with the new row selection values.

UNSELECT ALL ROWS

This Action will clear all rows from a table. This command will also clear all rows from any table that is contained within the target table.

UPDATE ROW

This command will cause the currently selected row in a table to be updated with the values in the Fields associated with the columns in the table.

Completing the Action

Click OK on the Code for Action <action name> dialog box, then click OK on the Action Properties dialog box to complete your Action definition.

Your new Action is added to the Action List.

Making an Action Context-Sensitive

Genesys Agent Scripting allows you to make an Action *context-sensitive*. For example, you can choose to show or hide a button (and therefore either allow or not allow its Action), based on the value of a particular Field.

Let's suppose that you have a `Check Box` field called `MoreInfo` that the script user should select if the customer requests additional information. If this box is selected, a `Select Info` button will be displayed allowing access to a new Page where the script user can select the type of information to send the customer. If not selected, the `Select Info` button will not be displayed.

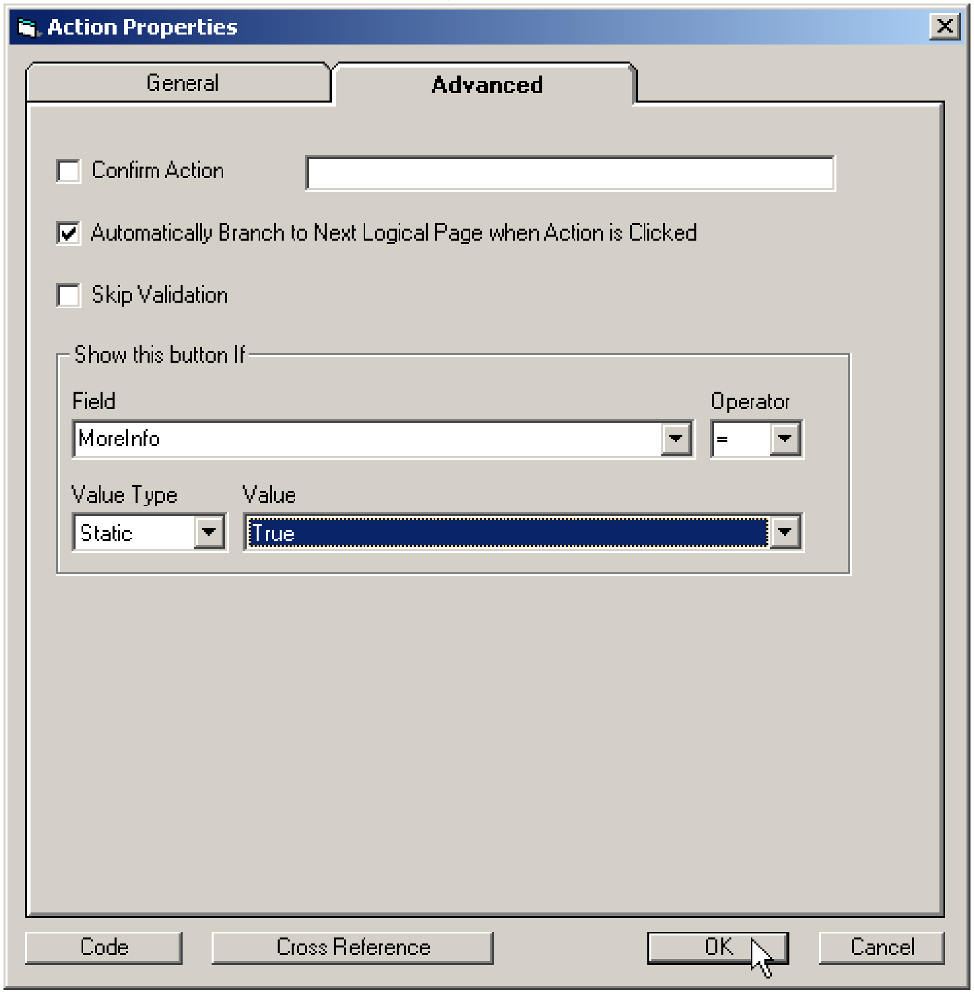
Set this up from the `Action Properties` dialog box for the `Select Info` action.

Procedure: Making an Action Context-Sensitive

Start of procedure

1. Select `Define` > `Actions` to open the `Action List` dialog box.
2. Select the `Select Info` action from the list and click `Edit` to open the `Action Properties` dialog box.
3. Select the `Advanced` tab on the `Action Properties` dialog box.

The Advanced tab is shown in Figure 53.



Action Properties

☐ Confirm Action

☑ Automatically Branch to Next Logical Page when Action is Clicked

☐ Skip Validation

Show this button If

Field: MoreInfo

Operator: =

Value Type: Static

Value: True

Figure 53: Making Select Info Context Sensitive

6. In the Show This Button If area, select MoreInfo from the Field dropdown list.

4. Select the “equals” = operator from the Operator dropdown list.

5. Select Static from the Value Type dropdown list.

6. Select True in the Value field.

   This statement now reads, essentially:

   "Show this button if the value of the MoreInfo field is True (if the MoreInfo Check Box is selected)."

7. Click OK to save the changes to the Select Info action button.

8. Click Close to close the Action List dialog box.

End of procedure

Genesys Agent Scripting allows a large amount of flexibility when setting up context-sensitive actions. You must, however, have already defined and created the Field on which the context-sensitive Action depends, since, as shown in Step 4 above, you select the Field from a dropdown list.

Examples

This section shows four step-by-step examples that illustrate how to create an Action:

- Creating a `Next Page` Action
- Creating an `Insert Row` Action for a customer data table
- Creating an Action using the `SET` Command to set one Field equal to the contents of another
- Creating an Alert to assure that a Field contains data before the script user advances to the next Page

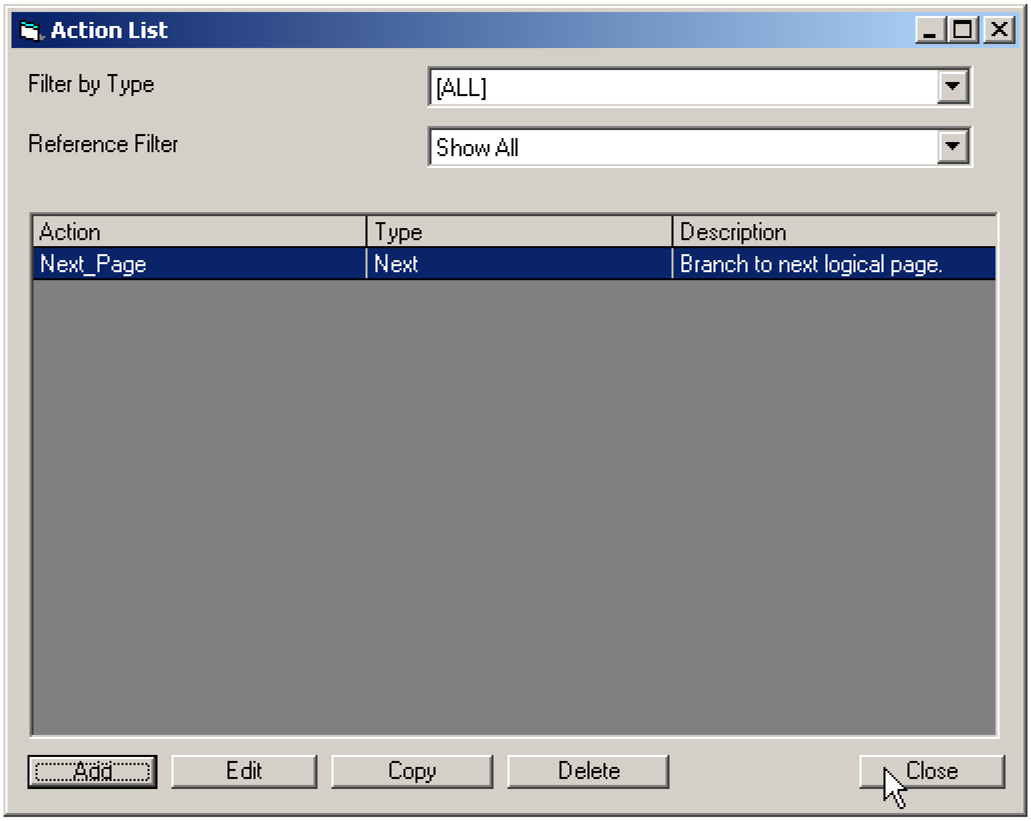
Procedure:
Creating a Next Page Action

Purpose: Create a **`Next Page`** action that can be used to navigate from one Page to the next.

Start of procedure

1. Select `Define` > `Actions`.
2. Click `Add` in the `Action List` dialog box.
3. In the `Action Wizard` dialog box, type a name for this Action in the `Action Name` field.
4. Select `Next` from the `Action Type` dropdown list.
5. Click `Finish`.
6. Unselect the `Edit Action` check box, then click `Close`.

   Your new Action is added to the `Action List` as shown in Figure 54 on page 143, which is normally displayed in the Genesys Agent Scripting Development Environment main window.



Filter by Type: [ALL]

Reference Filter: Show All

Action	Type	Description
Next_Page	Next	Branch to next logical page.

Figure 54: Action List

End of procedure

Creating an Insert Row Action

Procedure:
Create an Insert Row Action

Purpose: Create an Action that, when executed, will insert a new row into a table that holds customer name, address, and phone number information.

Note: This example requires a `Table` field that has been defined with customer information `String` fields. See the chapter called "Creating Fields"for details on how to create `String` and `Table` fields.

Start of procedure

1. Select Define > Actions.
2. Click Add in the Action List dialog box.
3. In the Action Wizard dialog box, type a name for this Action in the Action Name field.
4. Select Insert Row from the Action Type dropdown list.
5. Click Next.
6. In the Table dropdown list, select the table to which a row will be inserted when the Action is executed.
7. Click Finish.
8. Unselect the Edit Action check box, then click Close.
9. Your new Action is added to the Action List.

End of procedure

Creating a SET Command Action

Procedure:
Create a SET Command Action

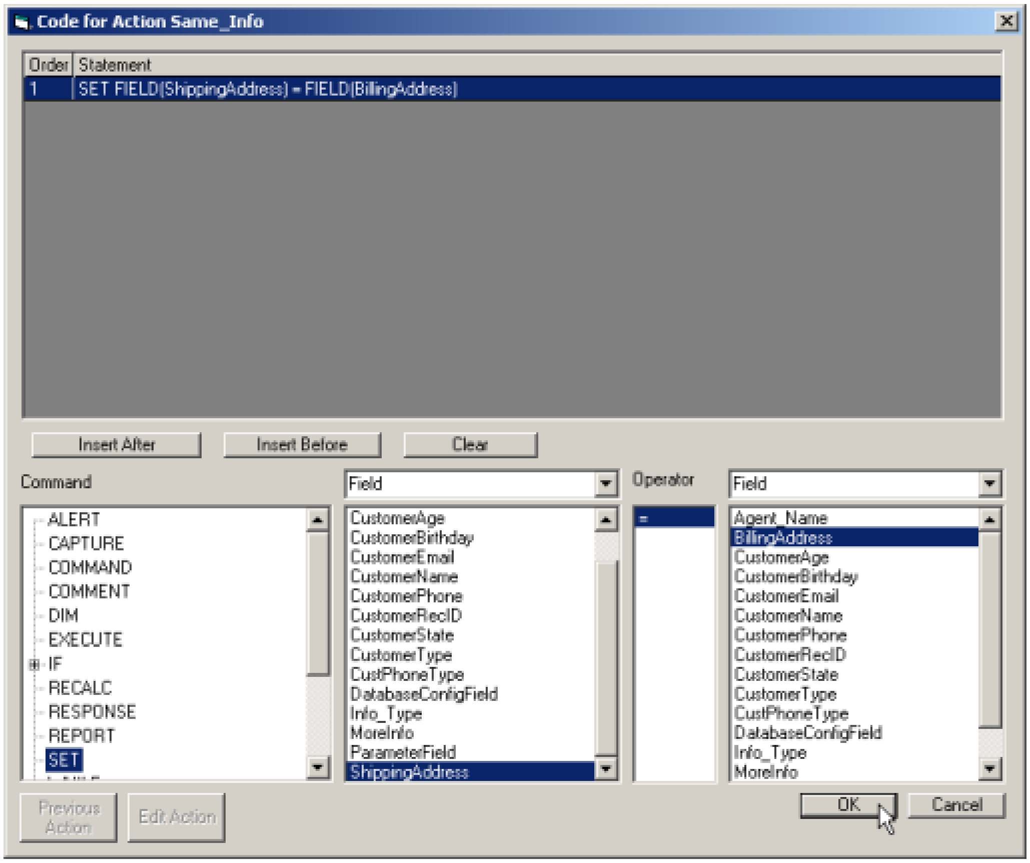
Our third example illustrates the use of a command to create an Action that sets the contents of one Field equal to another. When the Same Info button is clicked, the shipping address Field will be set equal to the contents of the billing address Field.

Note: This example requires String fields for both the billing address and the shipping address. See the chapter called "Creating Fields"for details on how to create String fields.

Start of procedure

1. Select Define > Actions.
2. Click Add in the Action List dialog box.
3. In the Action Wizard dialog box, type a name for this Action in the Action Name field.
4. Click Skip Wizard to open the Action Properties dialog box.
5. Type a value in the Name field, or Tab to it and Agent Scripting will enter a value for Name based on the contents of the Button Text field.
6. Enter an optional description for this Action in the Description field, if desired.

7. Make sure `Function` is selected in the `Action Type` field, then click `Code`.
8. Click `Insert After` in the `Code for Action <action name>` dialog box.
9. In the `Command` dropdown list, click `SET` as shown in Figure 55 on page 145.



Code for Action Same_Info

Order	Statement
1	SET FIELD(ShippingAddress) = FIELD(BillingAddress)

Command: SET
Field: ShippingAddress
Operator: =
Field: BillingAddress

Figure 55: Code for Action Dialog Box - SET Command

10. Select your shipping address Field from the first `Field` dropdown list.
11. Select the “equals” `=` in the `Operator` field.
12. Select your billing address Field from the second `Field` dropdown list.
13. Click `OK` to complete the code segment for the `SET` command.
14. Click `OK` in the `Action Properties` dialog box.

    Your new Action is added to the `Action List`.

End of procedure

Creating an Alert to Validate a Field

Our final example in this section illustrates the use of commands to create an Alert message to validate that a particular Field has been set before the script user moves on. If the script user attempts to advance to the next Page without entering a value for the customer name, an Alert message will be displayed to the script user.

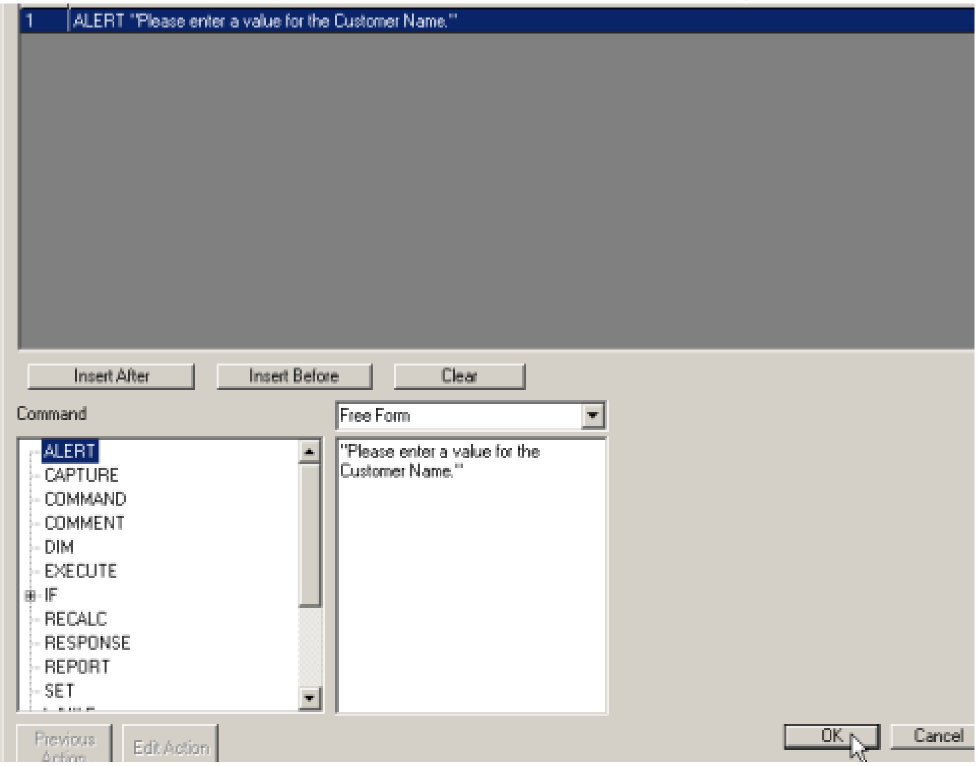
Note: This example requires a `String` fields that holds the customer's name. See the chapter called "Creating Fields"for details on how to create `String` fields.

Procedure:
Create an Alert Action

Start of procedure

1. Select `Define` > `Actions`.
2. Click `Add` in the `Action List` dialog box.
3. In the `Action Wizard` dialog box, type a name for this Action in the `Action Name` field.
4. Click `Skip Wizard` to open the `Action Properties` dialog box.
5. Type a value in the `Name` field, or `Tab` to it and Agent Scripting will enter a value for `Name` based on the contents of the `Button Text` field.
6. Enter an optional description for this Action in the `Description` field, if desired.
7. Make sure `Function` is selected in the `Action Type` field, then click `Code`.
8. Click `Insert After` in the `Code for Action <action name>` dialog box.

9. In the `Command` dropdown list, click `ALERT` as shown in Figure 56.



1 ALERT "Please enter a value for the Customer Name."

Command: Free Form

"Please enter a value for the Customer Name."

Figure 56: Code for Action Dialog Box - ALERT Command

10. Select `Free Form` from the dropdown list, since your alert will be a text message rather than the contents of a Field.
11. In the empty field below the dropdown list, type a message for this alert (such as "`Please enter a value for the Customer Name.`"). Be sure to include quotation marks.
12. Click `OK` to complete the code segment for the `ALERT` command.
13. Click `OK` in the `Action Properties` dialog box.

    Your new Action is added to the `Action List`.

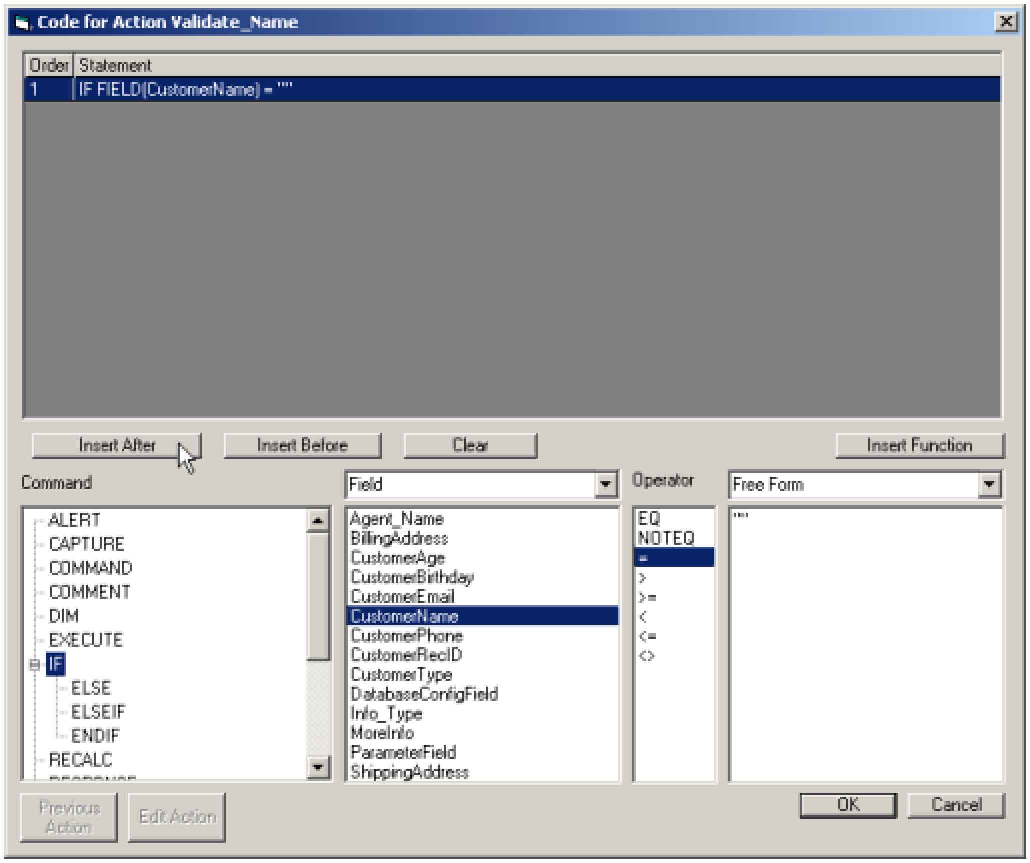
End of procedure

Procedure: Setting Up a Condition for the Alert

Purpose: You set up the Alert, but you still have to create an Action that determines when the Alert message should be displayed. If the script user attempts to move on to the next Page without entering a value for the customer name, then the Alert should be displayed. To set this up, you use `IF` and `EXECUTE` commands.

Start of procedure

1. Click `Add` in the `Action List` dialog box.
2. In the `Action Wizard` dialog box, type a name for this Action in the `Action Name` field.
3. Click `Skip Wizard` to open the `Action Properties` dialog box.
4. Type a value in the `Name` field, or `Tab` to it and Agent Scripting will enter a value for `Name` based on the contents of the `Button Text` field.
5. Enter an optional description for this Action in the `Description` field, if desired.
6. Make sure `Function` is selected in the `Action Type` field, then click `Code`.
7. Click `Insert After` in the `Code for Action <action name>` dialog box.
8. In the `Command` dropdown list, click `IF` as shown in Figure 57 on page 148.



Code for Action Validate_Name

Order	Statement
1	IF FIELD(CustomerName) = ""

Command: IF

Field: CustomerName

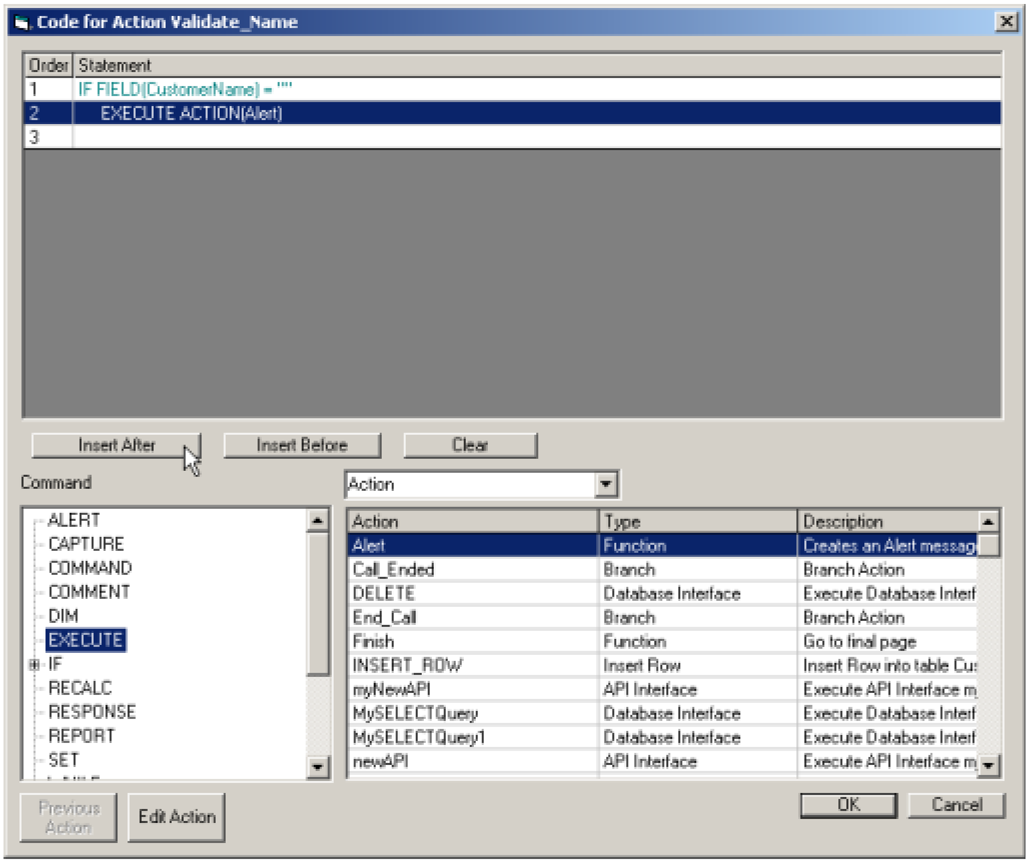
Operator: =

Free Form: ""

Figure 57: Code for Action Dialog Box - IF Command

9. Select your customer name Field from the first `Field` dropdown list.
10. Select “equals” `=` in the `Operator` field.
11. Select `Free Form` from the second dropdown list, since you are not comparing customer name to the value of another Field.

12. In the empty field below the dropdown list, type "" (meaning empty string).
13. Now you need to set what will happen if the condition (empty customer name Field) is met.
14. Click `Insert After` to set up the next command statement.
15. In the `Command` dropdown list, click `EXECUTE` as shown in Figure 58 on page 149.

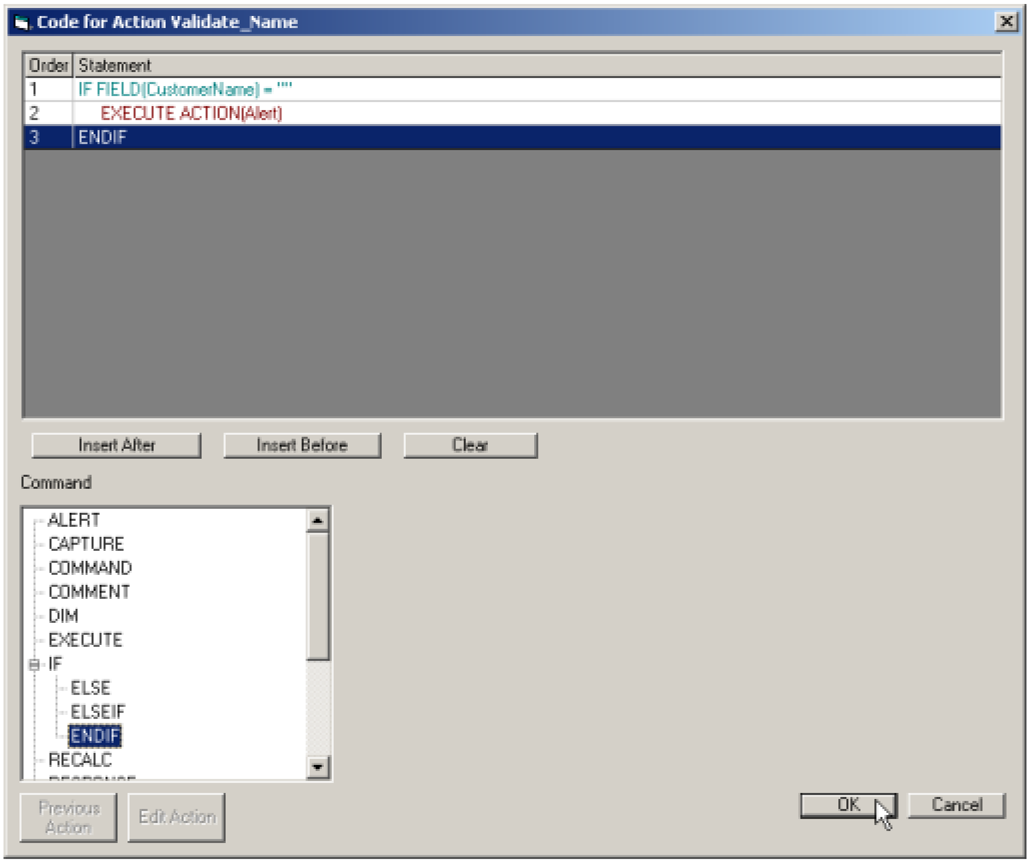


Order	Statement
1	IF FIELD(CustomerName) = ""
2	EXECUTE ACTION(Alert)
3	

Action	Type	Description
Alert	Function	Creates an Alert messag
Call_Ended	Branch	Branch Action
DELETE	Database Interface	Execute Database Interf
End_Call	Branch	Branch Action
Finish	Function	Go to final page
INSERT_ROW	Insert Row	Insert Row into table Cu
myNewAPI	API Interface	Execute API Interface m
MySELECTQuery	Database Interface	Execute Database Interf
MySELECTQuery1	Database Interface	Execute Database Interf
newAPI	API Interface	Execute API Interface m

Figure 58: Code for Action Dialog Box - EXECUTE Command

16. You will execute the `Alert` action you previously created, so select your `Alert` action from the `Action` dropdown list.
17. Click `Insert After` to add a new row to the code list so you can close the `IF` statement.
18. In the `Command` dropdown list, expand the Command tree for `IF` and click `ENDIF` as shown in Figure 59 on page 150.



Order	Statement
1	IF FIELD(CustomerName) = ""
2	EXECUTE ACTION(Alert)
3	ENDIF

Figure 59: Code for Action Dialog Box - ENDIF Command

19. Click `OK` to complete the `IF-EXECUTE-ENDIF` code segment.
20. Click `OK` in the `Action Properties` dialog box.

    Your new Action is added to the `Action List`.

End of procedure

Chapter

6 Using the Page Editor

This chapter provides an overview of the Page Editor, adding Pages to a Stream, and adding Fields and Actions to Pages.

The information in this chapter is divided among the following topics:

What Is the Page Editor?

A Page is the actual web page that appears on the agent desktop within a web browser. It contains all the text, Fields, Action buttons, Action code, interfaces, and branching conditions that have been set up for the Page.

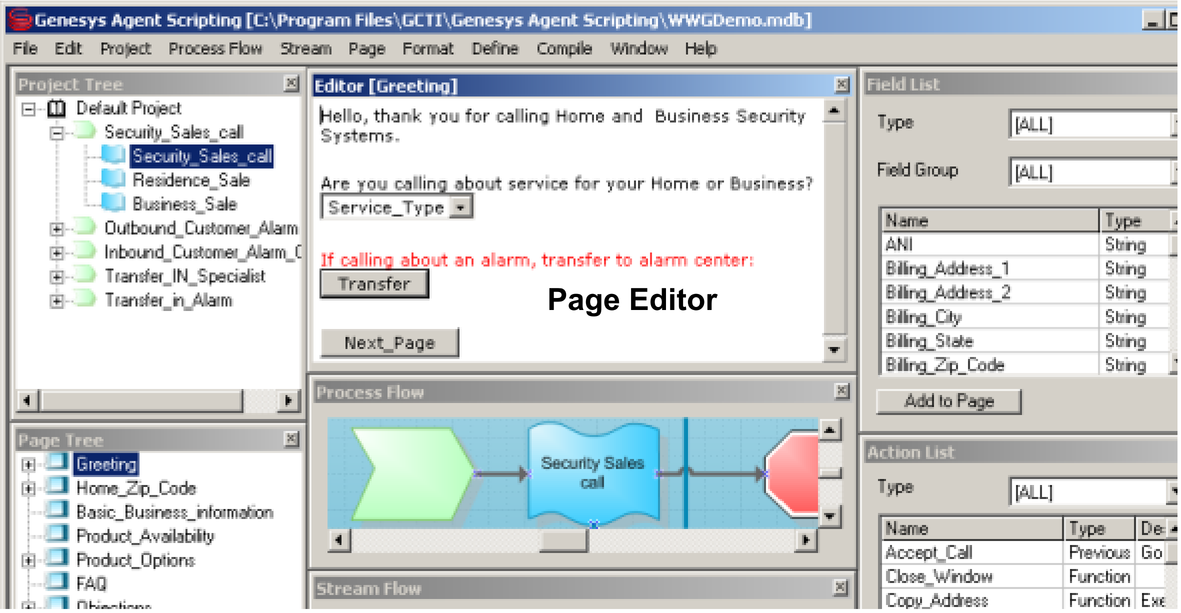
The Page Editor is the window inside the Genesys Agent Scripting Development Environment where you create and edit the individual web pages for your script application.

The Page Editor has two views:

- Page View
- HTML View

`Edit Page View` shows how the Page will be seen in a web browser. `Edit HTML` shows the underlying HTML code. You can toggle between these choices using `Page` > `Edit HTML` or `Page` > `Edit Page View`.

By default, the Page Editor appears in the center of the Agent Scripting main window as shown in Figure 60 on page 152.



Hello, thank you for calling Home and Business Security Systems.

Are you calling about service for your Home or Business?

If calling about an alarm, transfer to alarm center:

Field List: Type [ALL]; Field Group [ALL]

Name	Type
ANI	String
Billing_Address_1	String
Billing_Address_2	String
Billing_City	String
Billing_State	String
Billing_Zip_Code	String

Action List: Type [ALL]

Name	Type	De
Accept_Call	Previous	Go
Close_Window	Function	
Copy_Address	Function	Exe

Figure 60: The Genesys Agent Scripting Page Editor

Types of Objects You Can Add

You can add several types of objects to a Genesys Agent Scripting Page:

- Text
- Fields for data collection or display
- Action buttons that trigger execution of code
- Action code that can be executed when a Page loads or unloads
- Links that create text labels for branching to other script pages
- HTML tables
- URL links that create text labels for branching to other websites
- Navigator dropdown lists of Process Flows
- Images
- Breaks (line breaks without new paragraph)

Field and Action objects are defined as described in previous chapters and appear in the Field List and Action List in the Genesys Agent Scripting main window.

Creating Pages

There are several methods to create a new Page.

Procedure:
Create a New Page using the Define Menu

Start of procedure

1. Select Define > Pages to open the Page List dialog box as shown in Figure 61.

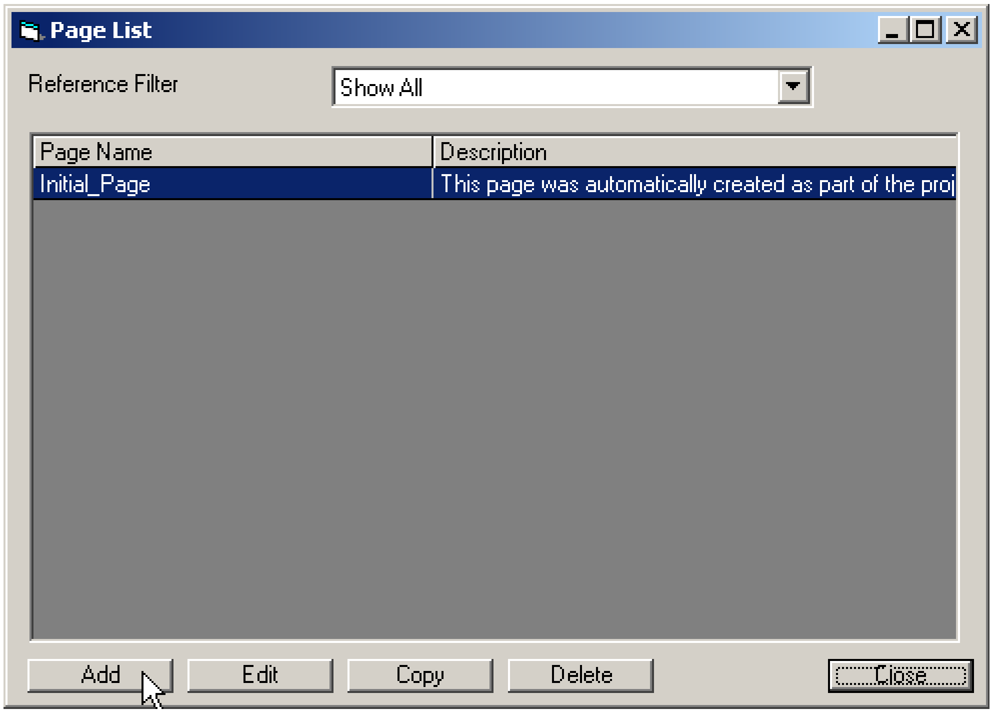
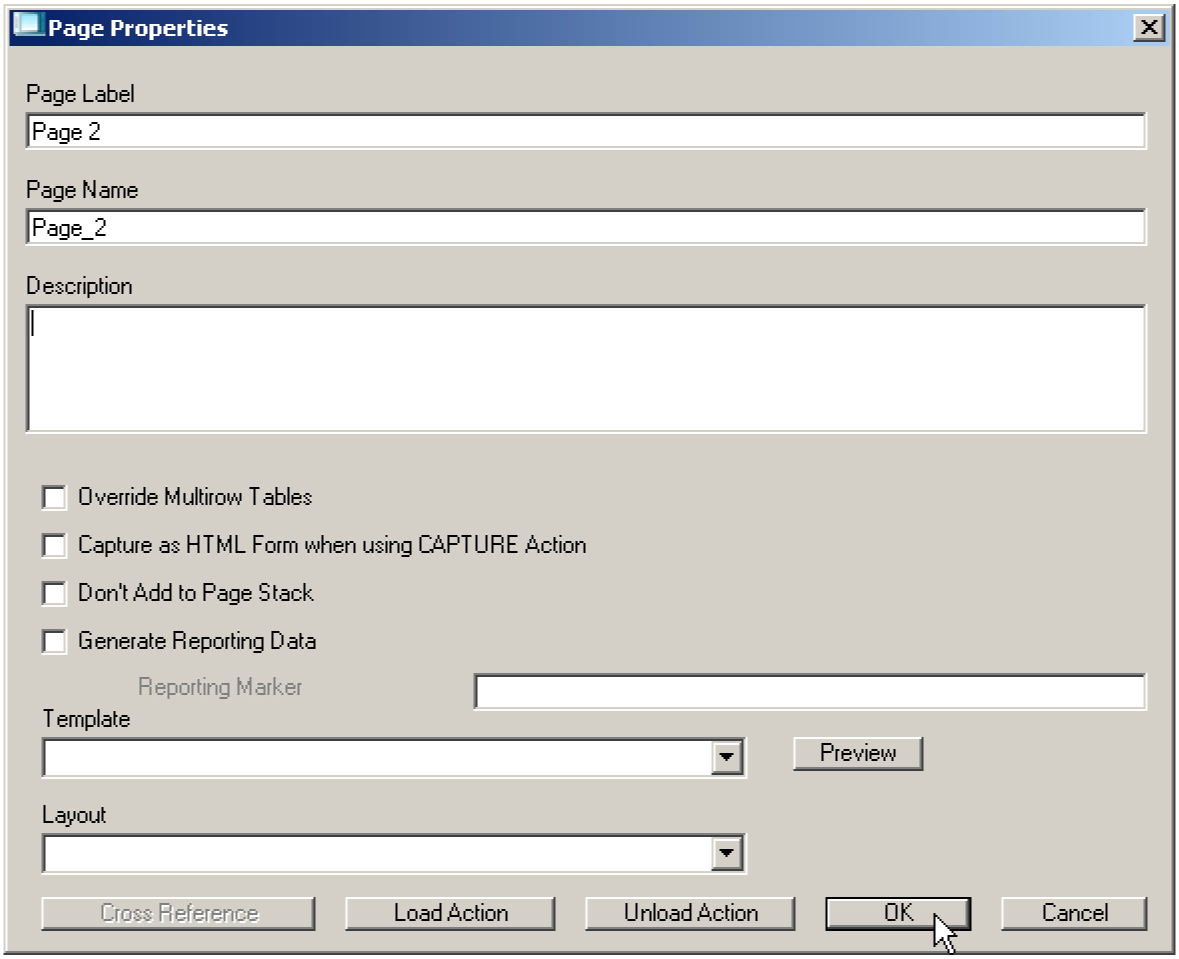


Figure 61: Page Properties Dialog Box

2. Click Add to initiate the process for creating a new Page.

   The Page Properties dialog box opens as shown in Figure 62 on page 154.



Page Properties

Page Label
Page 2

Page Name
Page_2

Description

□ Override Multirow Tables
□ Capture as HTML Form when using CAPTURE Action
□ Don't Add to Page Stack
□ Generate Reporting Data

Reporting Marker

Template

Layout

Figure 62: Page Properties Dialog Box

The `Page Properties` dialog box is where you define the characteristics that are unique to each Page you wish to create.

Table 28 describes the options on the `Page Properties` dialog box.

Table 28: Page Properties

Control	Description / What to Do...
Page Label	A text field that holds the value that describes this Page and is displayed at runtime. Page Labels appear in navigation boxes and in the Catalog. Type a label for the Page you are creating.
Page Name	A text field that holds the name that appears in and is referenced by the Agent Scripting Editor. Click this field or `Tab` to it to populate the field with a name that corresponds to the label you just entered.

Table 28: Page Properties (Continued)

Control	Description / What to Do...
Description	A text field that holds an optional description for this Page. (Optional) Type a description for this Page.
Override Multirow Tables	Tables are Fields defined as a collection of other Fields. Tables can either be defined as single row or multirow tables. Select this check box for all tables that appear on the script page to be treated as single row tables regardless of their definition.
Capture as HTML Form When Using CAPTURE Action	Select this check box to save the Page in HTML format rather than to just capture the content of the Page when using the `CAPTURE` action. This can be used to capture the layout of a letter or form and not just the contents or data on the form or letter.
Don't Add to Page Stack	Select this check box to keep a Page from being added to the page stack. This is useful for Pages that the script user executes but should not return to on a `Previous Page` action. For example, if there is a blank Page that only contains a `Page Load` event, that Page may be skipped if the script user backs up.
Generate Reporting Data	Select this check box to cause the fields of the page that are marked with `If Changed` or `Always` to be reported accordingly.
Reporting Marker	A text field that becomes available when `Generate Reporting Data` is selected. This field provides an additional dimension when collecting Page statistics and data. Refer to "Reporting" on page 375.

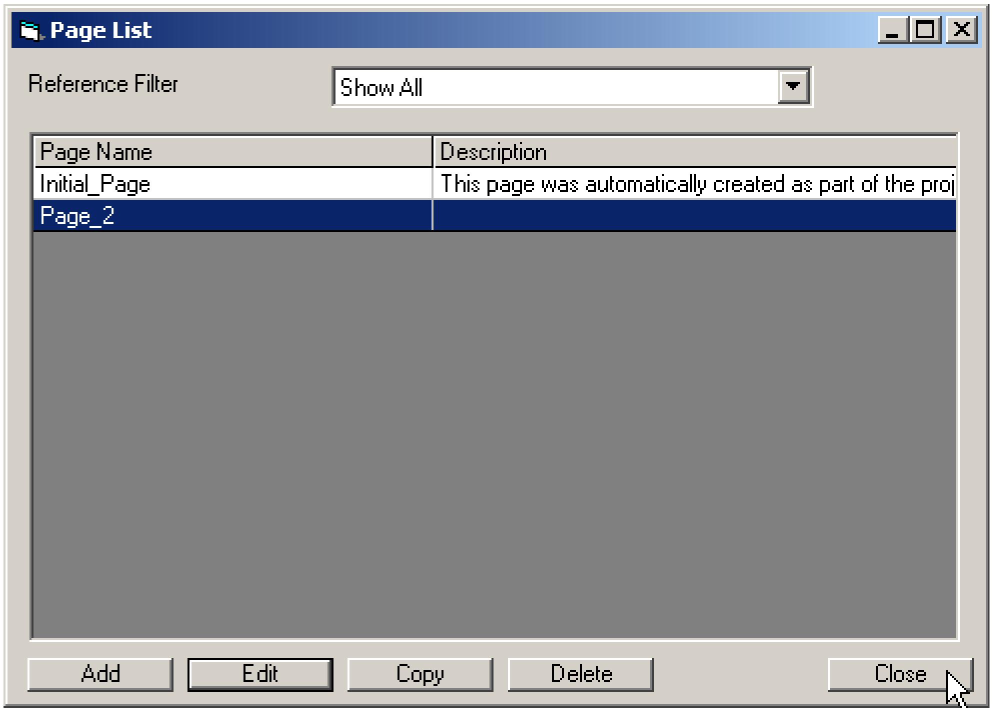
Table 28: Page Properties (Continued)

Control	Description / What to Do...
Template	A powerful feature of Genesys Agent Scripting is the use of templates for script Page Layouts. Using basic HTML structures, a template can be created in a text file, and then loaded into Genesys Agent Scripting. To format a script page based on a template, select the name of the template from the `Template` dropdown list. CAUTION: If you select a template to be applied to an existing script page, the Page's content will be replaced with the template.
Preview	Click `Preview` to open a web browser with the template loaded.
Layout	By default, Genesys Agent Scripting uses the Page Layout selected in the Settings dialog box as the layout for a Page. However, you can override the default Page Layout by selecting one from the dropdown list.
Cross Reference	This button becomes active once the Page has been created. Click `Cross Reference` to bring up a list of all of the objects that use the Page.
Load Action	Click `Load Action` to define an Action that will occur when this Page is loaded. The Action defined in `Load Action` occurs during runtime *prior to* the Page being rendered (displayed) in the script user's browser window. An example of a Load Action is an Action that initializes an input Field to a default value.

Table 28: Page Properties (Continued)

Control	Description / What to Do...
Unload Action	Click Unload Action to define an Action that will occur when this Page is about to unload. The Action defined in Unload Action occurs during runtime when a Page is unloaded, but *before* the Next Page logic is executed. An example of a Unload Action is an Action that validates user-provided input before moving to the next Page. If for any reason during this Action, the Alert command is encountered, the Page Unload is cancelled. This is useful for doing validation of information prior to leaving a Page.
OK	Click OK to create the new Page or save changes to the existing Page and close the Page Properties dialog box.

When you click OK in the Page Properties dialog box, the new Page is added to the Page List as shown in Figure 63.



Page Name	Description
Initial_Page	This page was automatically created as part of the proj
Page_2	

Figure 63: Page List Dialog Box - New Page Added

You can also create or edit Pages (and open the Page List dialog box), by selecting Page > Insert Page, or Stream > Insert Page, then clicking Add or Edit. In the latter case, the new Page you create will automatically be added to the selected Stream.

End of procedure

Procedure: Add a Page to a Stream

When you select a Stream in the Project Tree, you see the Pages that belong to that Stream in the Page Tree.

Creating a new Page does not necessarily result in the new Page being added to the desired Stream. The best way to assure that your new Page is part of the Stream is to add it manually after the Page is created.

Start of procedure

1. Select the Stream in the Project Tree.
2. Select Stream > Insert Page to open the Page List showing all defined Pages that are not currently part of this Stream.
3. Select the Page you wish to add to the Stream and click OK.
4. The Page Tree is updated to show the Page you selected.
5. Repeat this process for additional Pages.

End of procedure

Adding Content to Pages

The Page Editor functions in part as a word processor. You can add text by clicking and typing, you can arrange items by dragging and dropping, and you can cut and paste text or images.

As outlined earlier, you also add Fields, Actions, HTML tables, and other objects to Pages to create the functioning components of your script.

Procedure: Selecting a Page

Start of procedure

1. Click a Page in the Page Tree.

   The Page appears in the Page Editor.

End of procedure

Procedure: Adding Text to a Page

Start of procedure

1. Add text to a Page by positioning the cursor and typing.

   Straight text is used for instructions, Field labels, and other descriptive information.

End of procedure

Procedure: Adding Fields to a Page

In the chapter "Creating Fields" you learned how to define Fields. Available Fields appear in the `Field List` window in the Genesys Agent Scripting Development Environment.

Start of procedure

1. Select the Field in the `Field List` and click `Add to Page.`

End of procedure

Next Steps

Once a Field is on the Page, you can move the Field around on the Page and add some descriptive text.

Adding Actions to a Page

In the chapter "Creating Actions" you learned how to define actions. Available actions appear in the `Action List` window in the Genesys Agent Scripting Development Environment.

Actions can be represented as *buttons* that the script user clicks to execute the Action code.

Actions can also be tied directly to a Page, such that the Action code is executed when the Page loads or is about to unload.

Procedure: Add an Action button to a Page

Start of procedure

1. Select the Action in the `Action List` and click `Add to Page`.

End of procedure

Next Steps

Once an Action button is on the Page, you can move the button around on the Page and add some descriptive text.

Procedure: Add an Action that will execute when a Page loads or unloads:

Start of procedure

1. Right-click on your page in the `Page Tree` and select `Edit Page` to open the `Page Properties` dialog box for the Page to which you will tie the `Load` or `Unload` action.
2. Click `Load Action` or `Unload Action` as appropriate to open the `Action Code for Page Load <page name>` or `Action Code for Page Unload <page name>` dialog box.
3. Click `Insert After` to open the `Command` dropdown list.
4. Select the `EXECUTE` command.
5. Select the Action to execute when the Page loads (or unloads) from the `Action` dropdown list, then click `OK`.
6. Click `OK` to close `Page Properties.`

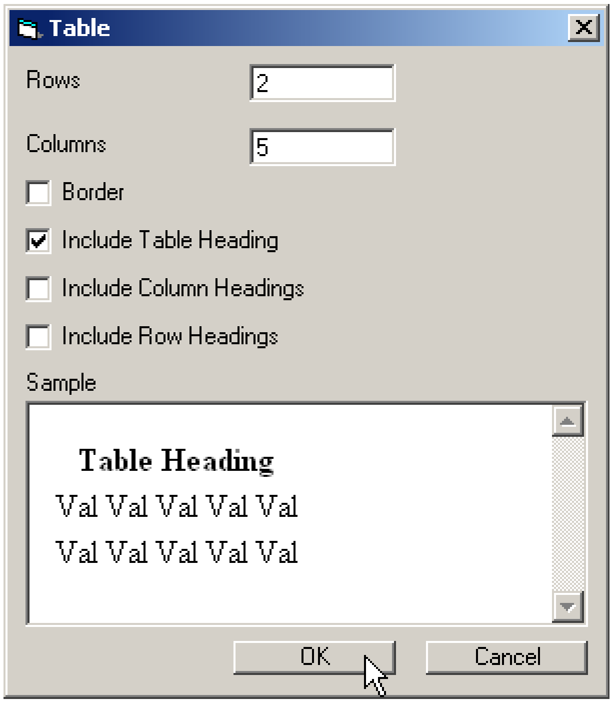
End of procedure

Procedure: Adding HTML Tables to a Page

Another object type you can add to a Page is an HTML table.

Start of procedure

1. Open the appropriate Page in the Page Editor.
2. Select `Page` > `Insert HTML Table` to open the `Table` dialog box as shown in Figure 64.



Rows 2

Columns 5

☐ Border

☑ Include Table Heading

☐ Include Column Headings

☐ Include Row Headings

Sample

Table Heading
Val Val Val Val Val
Val Val Val Val Val

Figure 64: Table Dialog Box

Table 29 describes the options and values on the `Table` dialog box.

Table 29: Table Dialog Box

Control	Description / What to Do...
Rows	Type the number of rows you want to be displayed in the table.
Columns	Type the number of columns you want to be displayed in the table.
Border	Select this check box to have a border for the table.
Include Table Heading	Select this check box to add a single row to the top of the table that spans all columns. This row is meant to be used as the heading for the table.
Include Column Headings	Select this check box to add a row to the table for column headings.
Include Row Headings	Select this check box to add a column to the table for row headings.
Sample	This field provides an example of what the table will look like on the Page.
OK	Click `OK` to add an HTML table with the selected characteristics to the Page.

End of procedure

Formatting HTML Tables

Once you add the HTML Table to your Page, select the HTML Table in the `Page Editor` and double-click to allow additional editing options.

This opens the `Edit HTML` dialog box, in which you can do one of the following:

- Edit the values in the table within the `Edit HTML` window, or
- Click `View HTML` to reveal the table's HTML code which you can modify as needed.

In addition, you can:

Procedure: Format and Modify the HTML Table

Start of procedure

1. Select an element in the HTML Table.
2. Select Format > HTML Table to reveal a list of HTML Table actions including the following:
   - Add Row
   - Delete Rows
   - Add Column
   - Merge Columns
   - Split Columns
   - Delete Columns
   - Add Cells
   - Delete Cells
   - Toggle Border

   The actions that are available depend on the element selected in the HTML Table.

   Note: The Add Column and Delete Columns menu items are disabled for tables that have a table heading.

3. Select the appropriate option from Format > HTML Table to make the desired changes to your table.

End of procedure

Editing HTML

The Page > Edit HTML menu option shows HTML code in the Page Editor for the currently selected Page or Page Layout. This enables you to edit the HTML code directly in any way you like. However, *you should only use this feature if you are experienced with HTML.*

Note: The compiler comments out custom server-side code that is written directly on the page; therefore, Genesys recommends that custom server-side scripts be written using an API Interface. The API Interface Properties dialog box has separate tabs for each target environment. For more information about API Interfaces, refer to Chapter 8, “API Interfaces,” on page 209.

Example

This section shows an example that illustrates how to create some Pages and add text, Fields, and actions to the sequence.

Procedure: Create the Pages

Start of procedure

1. Use `Define` > `Pages` to create a total of five new Pages.
2. Add each Page to the current Stream using `Stream` > `Insert Page`.
3. Reorder the Pages in the `Page Tree` by clicking and dragging individual Pages through the list.

End of procedure

Procedure: Create the Fields

- Refer to the chapter called "Creating Fields"and set up the Fields shown in Table 30.

Table 30: Fields for Example

Field Label/Name	Field Type	Details
Agent Name	String	N/A
CustomerName	String	N/A
CustomerPhone	String	`Size` tab, `Format` = `(###) ###-####`
MoreInfo	Check Box	N/A
BillingAddress	String	N/A
ShippingAddress	String	N/A
Info_Type	Radio Button	`Values` = `Brochure, Catalog, Order Form`

Creating the Actions

- Refer to the chapter called "Creating Actions"and set up the actions shown in Table 31.

Table 31: Actions for Example

Action Label/Name	Action Type	Details
Next	Next	Use Action wizard.
Previous	Previous	Use Action wizard.
Alert	Function	Command ALERT Free Form Value = "Please enter a value for the Customer Name."

Validate_Name	Function	Commands:
		IF FIELD(CustomerName)=""
		EXECUTE ACTION(Alert)
		ENDIF

Select_Info	Next	Use Action Properties: Select Automatically Branch when Action is Clicked Show this button if Field MoreInfo = True
Finish	Function	Command BRANCH [Current] Process Flow [Current] Stream Select Page from the dropdown list.
Same_Info	Function	Command SET Field ShippingAddress = Field BillingAddress

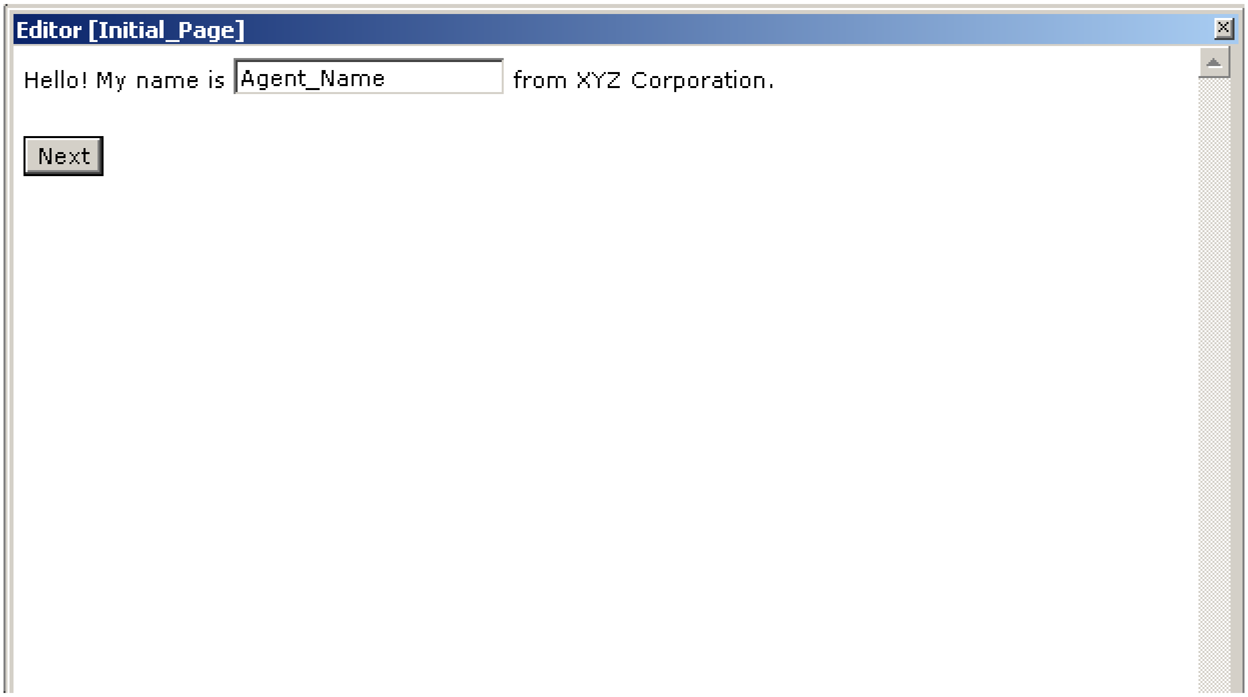
Procedure: Set Up Page 1

Start of procedure

1. Select the first page from the page tree to open it in the Page Editor.
2. Type Hello! My name is from XYZ Corporation.

3. Place the cursor between is and from, click Agent_Name in the Field List and then click Add to Page.
4. Press Enter to add a new line, click the Action Next in the Action List, and then click Add to Page.

Page 1 is shown in Figure 65 on page 166.



Hello! My name is Agent_Name from XYZ Corporation.

Figure 65: Page 1

End of procedure

Procedure: Set Up Page 2

Start of procedure

1. Select the second page from the page tree to open it in the Page Editor.
2. Type What is your name and phone number?
3. Press Enter to add a new line, click the Field CustomerName in the Field List, and then click Add to Page.
4. Press Enter to add a new line, click the Field CustomerPhone in the Field List, and then click Add to Page.
5. Press Enter to add a new line, click the Action Previous in the Action List, and then click Add to Page.

6. Click the Action `Next` in the `Action List`, and then click `Add to Page`.
7. Select `Page` > `Edit Page` to open `Page Properties`.
8. Click `Unload Action` to open `Action Code for Page Unload <page name>`.
9. Click `Insert After`.
10. Select `EXECUTE` from the `Command` dropdown list.
11. Select the Action `Validate_Name` from the `Action` dropdown list.
12. Click `OK`.
13. Click `OK` in `Page Properties`.

Note: The script user will need to enter a value in the `CustomerName` field before moving to the next Page. If not, the `Validate_Name` action will execute, and display the `Alert` message you set up earlier.

Page 2 is shown in Figure 66.

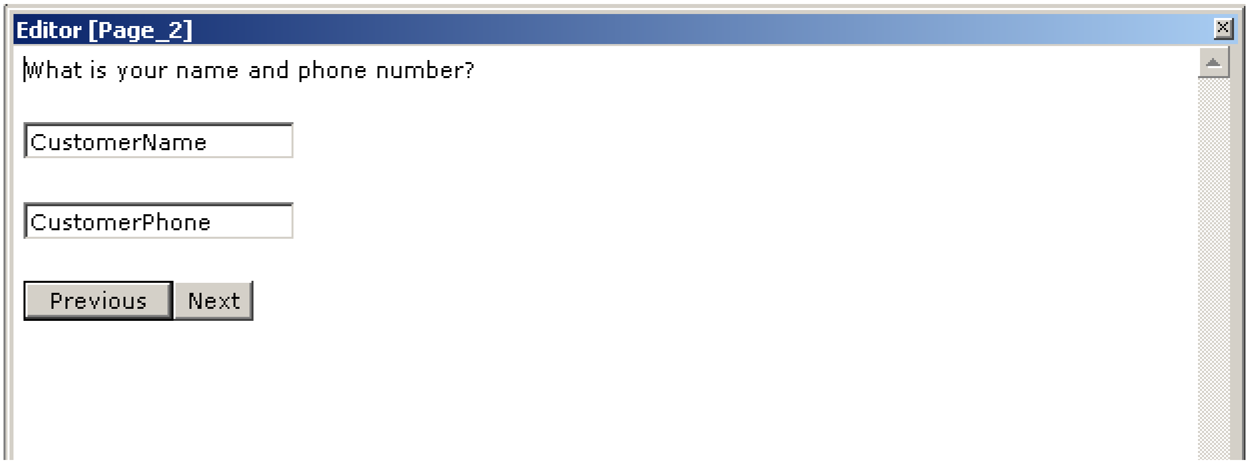


Figure 66: Page 2

End of procedure

Procedure:
Set Up Page 3

Start of procedure

1. Select the third page from the page tree to open it in the Page Editor.
2. Type `Would you like more information?`
3. Press `Enter` to add a new line, click the Field `MoreInfo` in the `Field List`, and then click `Add to Page`.

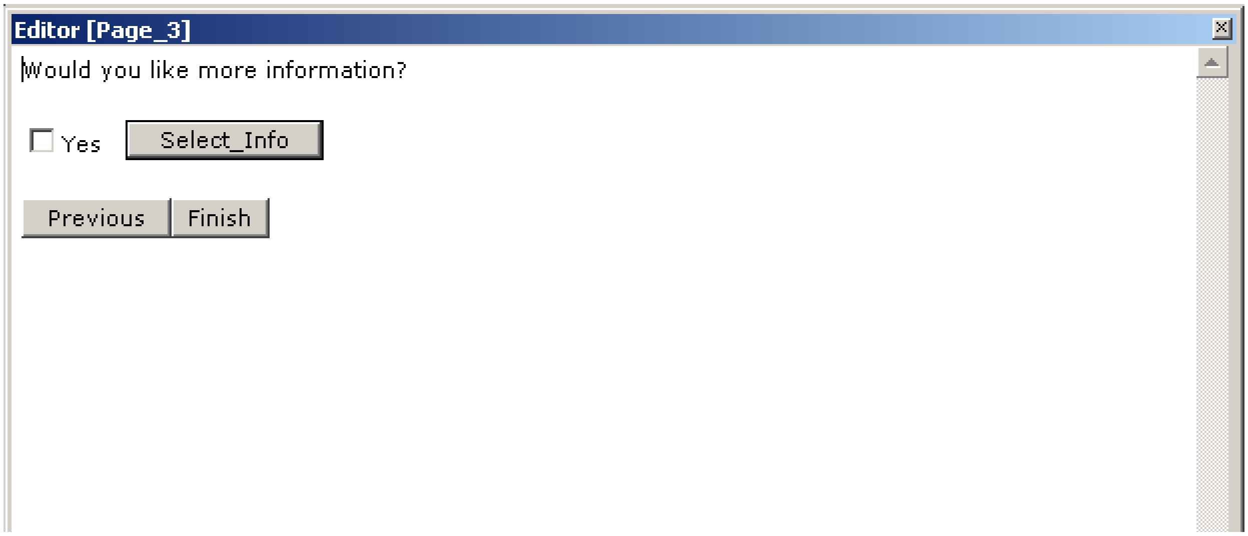
4. Type Yes beside the check box, then add a few spaces.
5. Click the Action Select_Info in the Action List, and then click Add to Page.

Note: The Select_Info action is defined as *context-sensitive*. The button will only display during execution of the script if the MoreInfo check box is selected.

6. Press Enter to add a new line, click the Action Previous in the Action List, and then click Add to Page.
7. Click the Action Finish in the Action List, and then click Add to Page.

Note: The Finish action is defined to branch to the final Page (5 of 5). It skips the selection of information to come on the next Page and is used when MoreInfo is not selected. The Next action is not added to this Page because Select_Info does an automatic branch when the Action is clicked.

Page 3 is shown in Figure 67.


Editor [Page_3]

Would you like more information?

Figure 67: Page 3

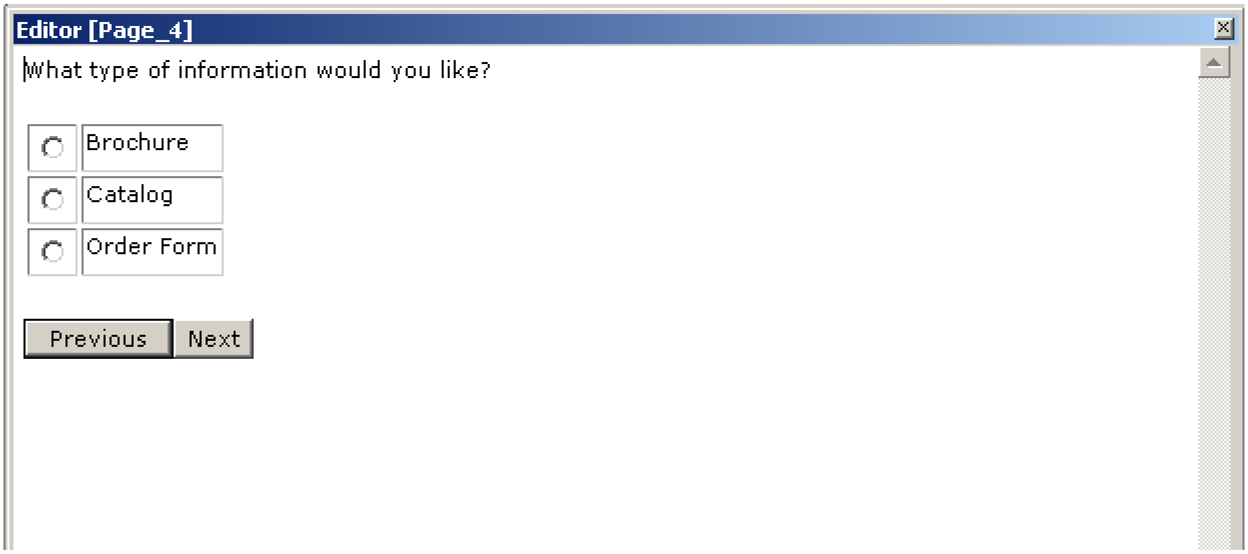
End of procedure

Procedure: Set Up Page 4

Start of procedure

1. Select the fourth Page from the `Page Tree` to open it in the Page Editor.
2. Type `What type of information would you like?`
3. Press `Enter` to add a new line, click the Field `Info_Type` in the `Field List`, and then click `Add to Page`.
4. Press `Enter` to add a new line, click the Action `Previous` in the `Action List`, and then click `Add to Page`.
5. Press `Enter` to add a new line, click the Action `Next` in the `Action List`, and then click `Add to Page`.

Page 4 is shown in Figure 68 on page 169.



Editor [Page_4]

What type of information would you like?

○ Brochure
○ Catalog
○ Order Form

Figure 68: Page 4

End of procedure

Procedure: Set Up Page 5

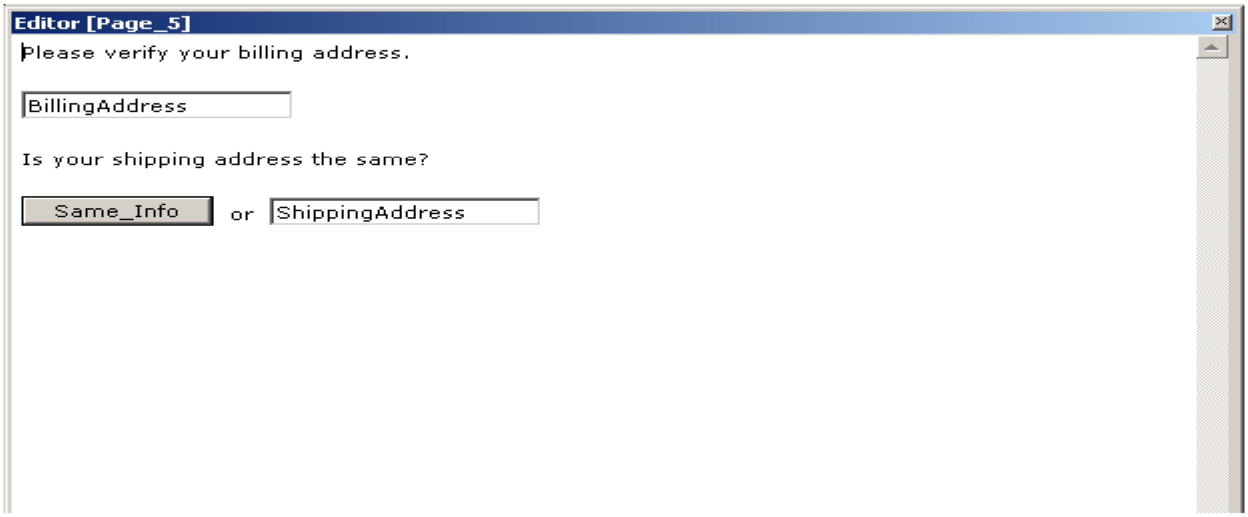
Start of procedure

1. Select the fifth page from the page tree to open it in the Page Editor.
2. Type `Please verify your billing address.`

3. Press Enter to add a new line, click the Field BillingAddress in the Field List, and then click Add to Page.
4. Press Enter to add a new line, then type Is your shipping address the same?
5. Press Enter to add a new line, click the Action Same_Info in the Action List, and then click Add to Page.
6. Add a few spaces, type or, then add a few more spaces.
7. Click the Field ShippingAddress in the Field List, and then click Add to Page.

Note: The Same_Info action contains code that sets the shipping address equal to the contents of the billing address Field. The script user clicks this button if the customer answers Yes to the question in Step 4. If the answer is No, the script user can enter a unique shipping address.

Page 5 is shown in Figure 69 on page 170.



Please verify your billing address.

Is your shipping address the same?

Figure 69: Page 5

End of procedure

Compile and Test

When you have a script segment that you wish to test, such as the example we've just completed above, you should compile the code and run Simulate to see how things work.

The procedures for compiling and testing (simulating) your code are described in Chapter 12, "Compiling, Testing, and Deploying Your Application," on page 269.

Genesys

Chapter

7 Databases and Database Interfaces

This chapter provides an overview of how to use databases and Database Interfaces within Genesys Agent Scripting.

The information in this chapter is divided among the following topics:

Why Use Databases?

An earlier chapter introduced the concepts of Fields, which allow script users to enter data manually, and Actions, which allow script users to set the value of a Field.

Sometimes important data is already available in an external database and the Agent Scripting application needs to access and use that information.

Agent Scripting can read information from a database table as long as a *connection* is created that tells the application how to access the data. This is defined in two parts:

- Database
- Database Interface

The *Database* tells what the Database connection strings are.

The *Database Interface* tells what type of query is being made and the mapping of the database fields with Agent Scripting Fields. Both the Database and Database Interface are needed to make a Database connection.

Supported Database Formats

Genesys Agent Scripting can connect to the following types of databases:

- Oracle
- SQLServer
- DB2

What Is a Database Connection?

For an Agent Scripting application to connect to an external database, an Agent Scripting Database connection must be defined. The Database connection contains the external database name, and the type of database and connection strings for development and runtime connections. This information allows the application to locate and access the external database.

What Is a Database Interface?

In order to retrieve information that is stored in an external database, or update the information that is stored in a database table, you need to create a Database Interface in the Agent Scripting Development Environment.

A Database Interface functions as a *query* to a specific external database. Genesys Agent Scripting supports three types of queries:

- **Select** — Retrieves information from an external database.
- **Update** — Inserts new rows, updates existing rows, or deletes rows in a database table. In order to insert rows, your database will need to have a key column defined.
- **Stored Procedure** — Passes values as input parameters to a Stored Procedure. You can also define Fields to store output parameters and results (return parameters) from a Stored Procedure. For more information on Stored Procedures, refer to “Creating a Stored Procedure” on page 330.

Creating a Database Connection

Procedure:
Create a New Database Connection

You will use the Define menu.

1. Select `Define` > `Databases` to open the `Database List` dialog box as shown in Figure 70.

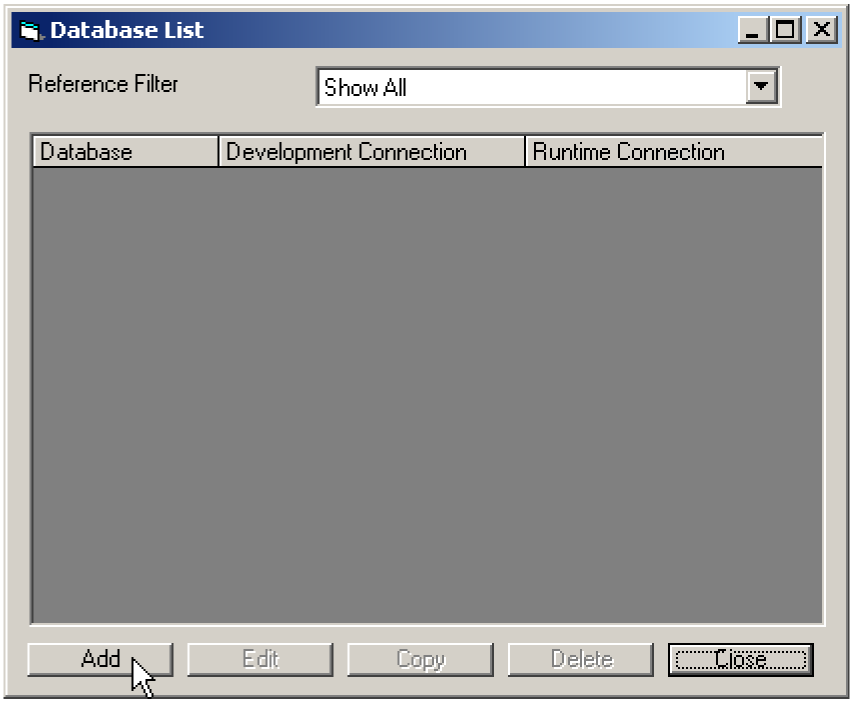
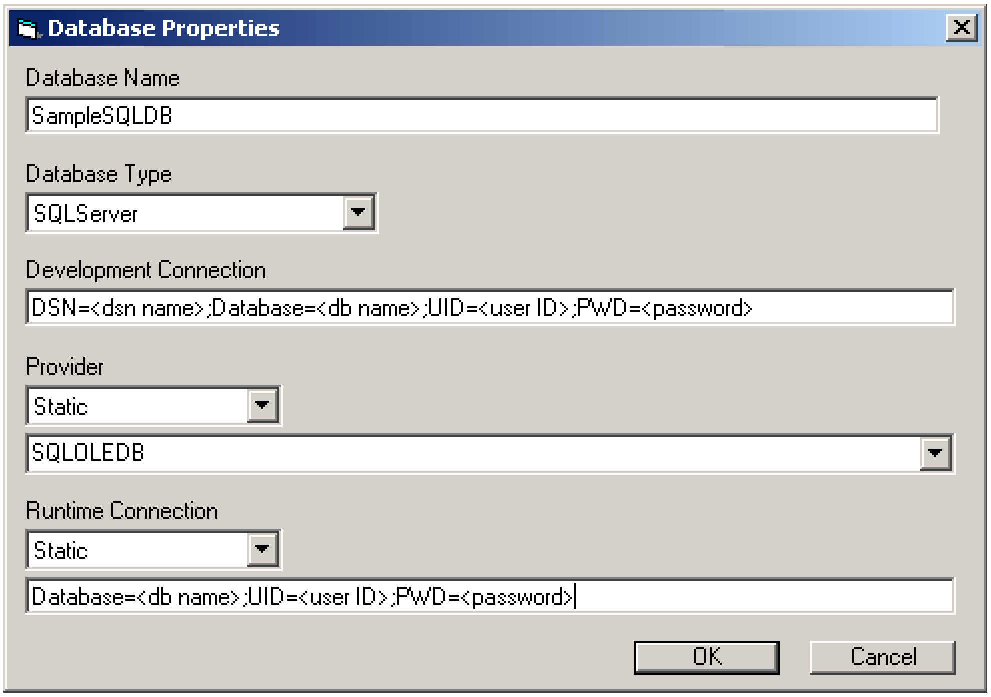


Figure 70: Database List Dialog Box

2. Click `Add` to initiate the process for creating a new Database connection.

The `Database Properties` dialog box opens as shown in Figure 71 on page 174.



Database Name: SampleSQLDB

Database Type: SQLServer

Development Connection

DSN=<dsn name>;Database=<db name>;UID=<user ID>;PWD=<password>

Provider: Static

SQLOLEDB

Runtime Connection: Static

Database=<db name>;UID=<user ID>;PWD=<password>

Figure 71: Database Properties Dialog Box - SQLServer for ASP

`Database Properties` allows you to provide all the information required to set up a Database connection. Some of the values you provide or select in the `Database Properties` dialog box will differ depending on your Target Environment.

Table 32 describes the options on the `Database Properties` dialog box.

Table 32: Database Properties

Control	Description / What to Do...
Database Name	The value in this field is a user-defined reference that Genesys Agent Scripting uses to refer to the database. This name is used when selecting a database as the data source for the Database Interface.
Database Type	Select the Database type from the dropdown list. This represents the format of the database for which you're building the connection. Available choices are `Oracle, SQLServer, DB2, and Other`.

Table 32: Database Properties (Continued)

Control	Description / What to Do...

Development Connection	The `Development Connection` field specifies a connection string that the Genesys Agent Scripting Development Environment uses to connect to the target database. This connection is used to allow Genesys Agent Scripting to retrieve metadata (such as tables and columns) that describe the target database.
	For Oracle databases, the format is:
	`DSN=<dsn name>;UID=<user ID>;PWD=<password>`
	For SQLServer databases, the format is:
	`DSN=<dsn name>;Database=<db name>;UID=<user ID>;PWD=<password>`
	`<dsn name>` = the Data Source Name for this database type, which you can find or create in Windows under `Control Panel` > `Administrative Tools` > `Data Sources (ODBC)`.
	`<db name>` = the actual name of the database file.
	`<user ID>` = login ID, which may be your Windows login or your database server login, depending on the authentication method selected for the database.
	`<password>` = login password, which may be your Windows password or your database server password, depending on the authentication method selected for the database.
	An example follows of a Development Connection string for an SQLServer database type where the DSN is defined as `SQLServerDSN`, the name of the database is `SampleSQLDB`, and Windows login/password is `johnsmythe`, `astro22`:
	`DSN=SQLServerDSN;Database=SampleSQLDB;UID=johnsmythe;PWD=astro22`

Table 32: Database Properties (Continued)

Control	Description / What to Do...
Provider	The `Provider` field is used at runtime to tell Agent Scripting which driver to use to talk to the database. This information is provided by the database manufacturer. The Provider string you need depends on the database type *and* the Target Environment. If you select `Static` in the `Provider` dropdown list, the `Provider` field will display string choices appropriate to the database type selected in the `Database Type` dropdown list. If you select `Field` in the `Provider` dropdown list, the `Provider` field will display a list of all Fields. Select the Field that holds the Provider string. See Table 33 on page 177 for some available string values.
Runtime Connection	The `Runtime Connection` field specifies a connection string to be used by the generated Genesys Agent Scripting application at runtime. If you are unsure of the values to use in the `Runtime Connection` and `Development Connection` fields, speak with your database administrator. A runtime connection can be statically defined or have its value stored in and read from a Field so that it can be dynamically changed at run time. If you select `Static` in the `Runtime Connection` dropdown list, type a runtime connection string in the `Runtime Connection` field. If you select `Field` in the `Runtime Connection` dropdown list, the `Runtime Connection` field will display a list of all Fields. Select the Field that holds the runtime connection string. **Note:** Database connection strings can vary depending on database configuration and options, security requirements, and other factors. Genesys Agent Scripting provides samples and suggestions for the appropriate connection string based on the selected provider. However, you should contact your system or database administrator for the correct database connection string for your environment.
OK	Click `OK` to complete the Database connection. The connection is added to the `Database List.`

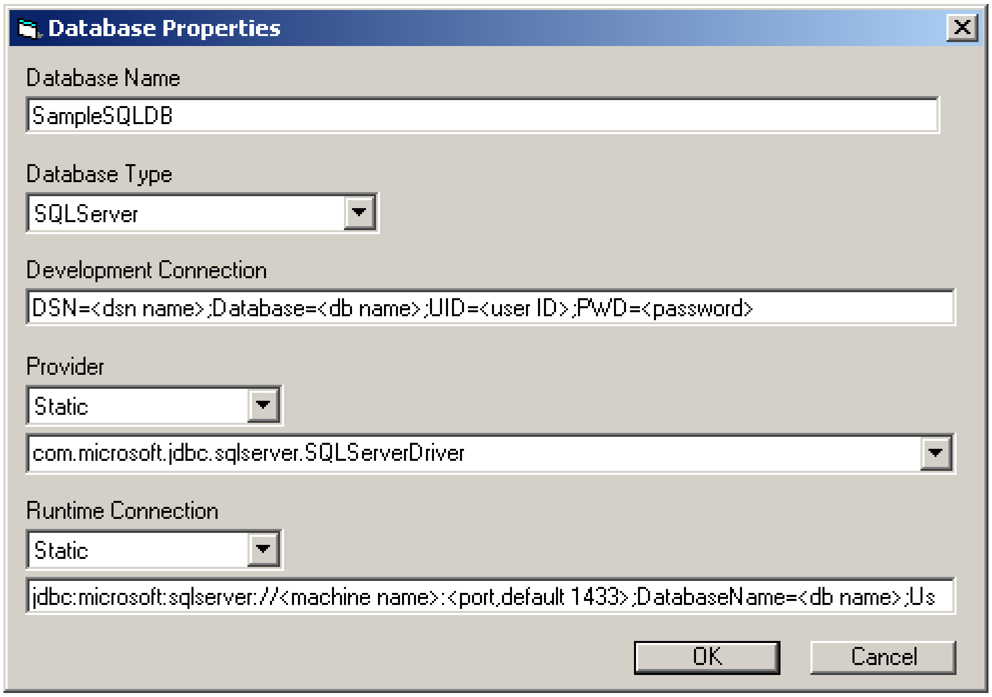
Table 33: Providers

Database Type	ASP	ASPX	JSP
Oracle	OraOLEDB.Oracle	OraOLEDB.Oracle and System.Data.OracleClient	oracle.jdbc.OracleDriver (This is the standard Provider string from Oracle.) and oracle.jdbc.pool.OracleConnectionCacheImpl (This offers improved performance by caching Database connections in Agent Scripting's application scope on the web server.)
SQLServer	SQLOLEDB	SQLOLEDB and System.Data.SqlClient	com.microsoft.sqlserver.jdbc.SQLServerDriver
DB2	IBMDADB2	IBMDADB2 and IBM.Data.DB2	com.ibm.db2.jcc.DB2Driver

Refer back to Figure 71 on page 174 to see a Database Properties example (with placeholders in Development Connection and Runtime Connection) for an SQLServer database named SampleSQLDB with an ASP scripting application deployment.

Figure 72 shows a Database Properties example (again with placeholders in Development Connection and Runtime Connection) for an SQLServer database named SampleDQLDB with a JSP scripting application deployment.

.



Database Name
SampleSQLDB

Database Type
SQLServer

Development Connection
DSN=<dsn name>;Database=<db name>;UID=<user ID>;PWD=<password>

Provider
Static
com.microsoft.jdbc.sqlserver.SQLServerDriver

Runtime Connection
Static
jdbc:microsoft:sqlserver://<machine name>:<port,default 1433>;DatabaseName=<db name>;Us

Figure 72: Database Properties Dialog Box - SQLServer for JSP

Creating a Database Interface

Procedure:
Create a New Database Interface

You will use the `Define` menu.

Start of procedure

1. Select `Define` > `Database Interfaces` to open the `Database Interface List` dialog box as shown in Figure 73.

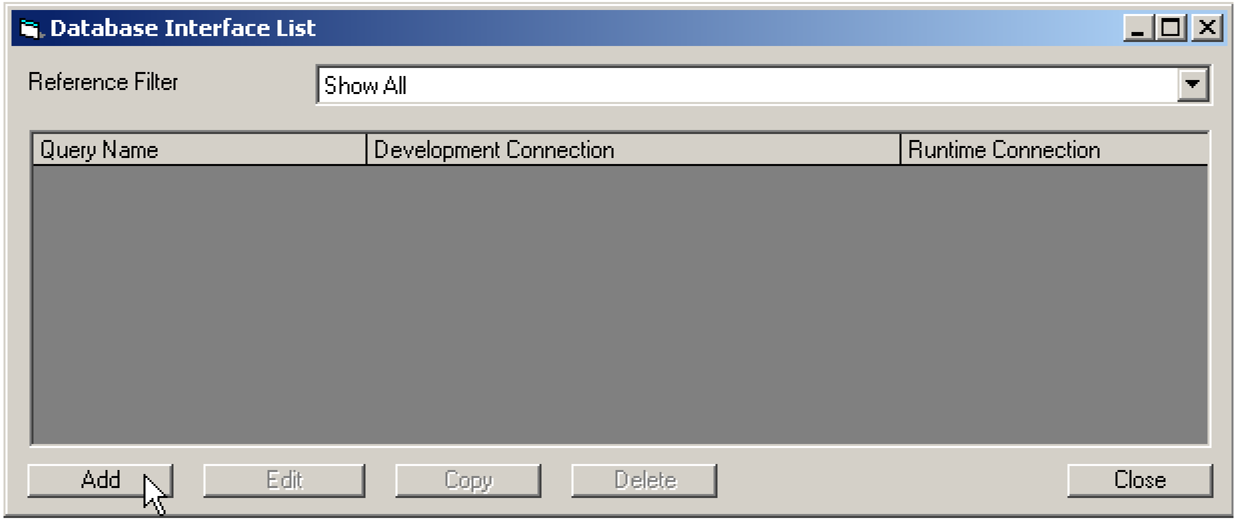
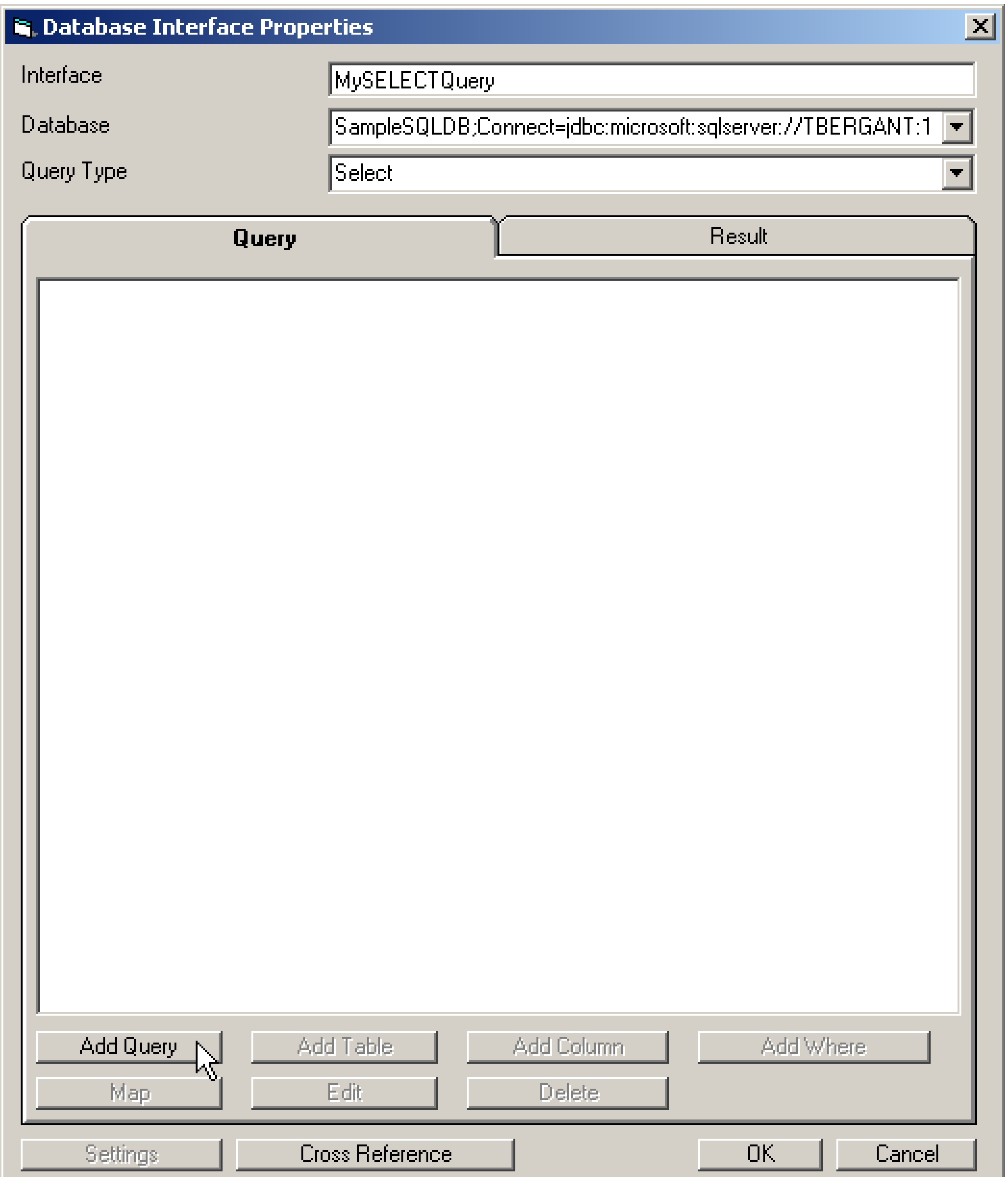


Figure 73: Database Interface List Dialog Box

2. Click `Add` to initiate the process for creating a new Database Interface.

   The `Database Interface Properties` dialog box opens as shown in Figure 74 on page 180.



Database Interface Properties

Interface: MySELECTQuery

Database: SampleSQLDB;Connect=jdbc:microsoft:sqlserver://TBERGANT:1

Query Type: Select

Figure 74: Database Interface Properties Dialog Box

`Database Interface Properties` allows you to set up a query (of type Select, Update, or Stored Procedure) to an external database.

Table 34 on page 181 describes the options on the `Database Interface Properties` dialog box.

Table 34: Database Interface Properties

Control	Description / What to Do...
Interface	Type a name for this Database Interface. A name descriptive of the type of query may be most helpful.
Database	As long as you have previously set up a Database connection as described in “Creating a Database Connection” on page 173, select the Database connection in the `Database` dropdown list. The dropdown list displays all defined Genesys Agent Scripting Database connections in the following format: `xxx;Connect=yyy` `xxx` = the name of the Database connection `yyy` = the runtime connection string specified for the Database.
Query Type	Select the query type in the Query Type dropdown list. Supported query types are: • Select • Update • Stored Procedure
Query Tab	Buttons on the `Query` tab provide the means by which you develop the details of the query. The query is displayed in the large field below the tab. See “Building a Query” on page 183 for details on building queries.
Add Query	After naming the interface, selecting the Database connection, and selecting the query type, click `Add Query` to start building a query appropriate to your selected query type. See “Add Query” on page 183 for details.
Add Table	Once you’ve begun your query definition, click `Add Table` to identify the schema, database table, and key column for your query. See “Add Table” on page 185 for details.
Add Column	After adding the table, click `Add Column` to identify the columns in the database table for your query. See “Add Column” on page 187 for details.

Table 34: Database Interface Properties (Continued)

Control	Description / What to Do...
Add Where	This option is used to place conditions on the query. You can have from zero to many Where clauses for a query. See "Add Where" on page 190 for details.
Map	After adding tables and columns to your query, click Map to organize your query results by mapping query elements to Genesys Agent Scripting Fields. You can accomplish the same mapping using the Result tab (see below). However, the Map operation allows you to do this from a single dialog box. See "Using the Map Button" on page 198 for details.
Edit	Select the Query, Table, or Column in the Query tab and click Edit to make modifications to the selected query element.
Delete	Select the Query, Table, or Column in the Query tab and click Delete to remove the selected query element.
Result Tab	Results from a database query are defined based on the Query tab. Results are returned to Genesys Agent Scripting in an XML format. Use the Result tab to map query results to Agent Scripting Fields. You cannot add or delete tags. Tags are created based on the query definition. See "Using the Result Tab" on page 193 for details on setting up and organizing query results.
Edit	Select the result in the Result tab and click Edit to make modifications to the selected result element.
Cross Reference	The Cross Reference button allows you to see where a Database Interface is being used. Click Cross Reference to open the Object Cross Reference dialog box.
OK	Click OK to complete setup of the Database Interface. The new Database Interface is added to the Database Interface List.

3. Save your changes.

End of procedure

Building a Query

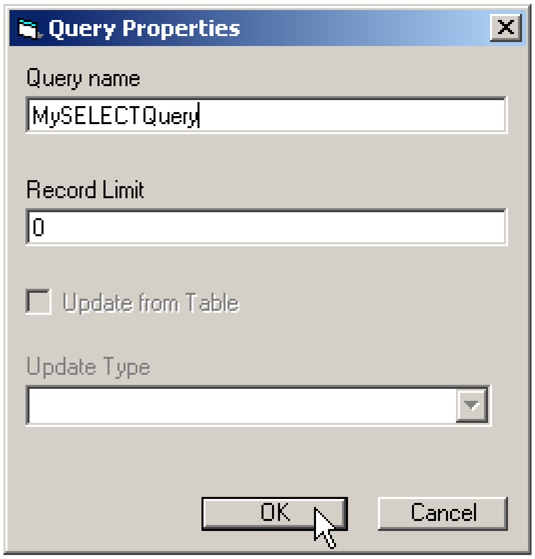
Building a query involves several steps, each represented by a button on the Query tab of Database Interface Properties:

- Add Query
- Add Table
- Add Column
- Add Where (optional)

Procedure: Add Query

Start of procedure

1. Click Add Query to open the Query Properties dialog box as shown in Figure 75.



Query name: MySELECTQuery

Record Limit: 0

Figure 75: Query Properties Dialog Box

The Query Properties dialog box is where you define some basic details regarding your query.

Table 35 describes the options in the Query Properties dialog box.

Table 35: Query Properties

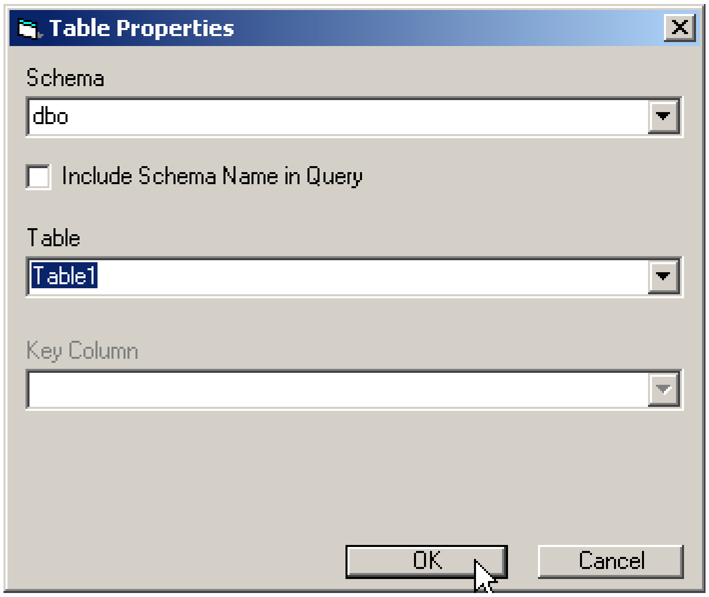
Control	Description / What to Do...
Query Name	Type a name for this Query. A name descriptive of the type of query may be most helpful. For a stored procedure, the Query name must be the name of the stored procedure.
Record Limit	(Available for Select and Stored Procedure queries only.) An integer limits what's returned to that number of matching records. A `Record Limit` of `0` means that all records that match the query parameters will be returned.
Update from Table	(Available for Update queries only.) Select this check box to indicate that values for the query will come from a table. When this option is selected a `Field` dropdown list becomes active and replaces the `Update Type` dropdown list.
Field	(Available for Update queries only, when the `Update from Table` check box is selected.) Select the `Table` field that will be used for this update in the `Field` dropdown list. Each row in a table has a built in indicator managed by Genesys Agent Scripting that will structure the query for Inserts, Updates, and Deletes based on the actions of the users.
Update Type	(Available for Update queries only, when the `Update from Table` check box is not selected.) Select an update type for this query. The valid choices are: • Update • Insert • Delete
OK	Click `OK` when query properties have been set. Agent Scripting returns you to the `Database Interface Properties` dialog box where you define more details about your query.

End of procedure

Procedure: Add Table

Start of procedure

1. Select the Query in the `Query` tab and click `Add Table` to open the `Table Properties` dialog box as shown in Figure 76.



Schema: dbo

□ Include Schema Name in Query

Table: Table1

Key Column:

Figure 76: Table Properties Dialog Box

The `Table Properties` dialog box is where you select the database schema and the database table affected by your query. For Update queries, you also specify the Key Column in the database table.

Table 36 describes the options in the `Table Properties` dialog box.

Table 36: Table Properties

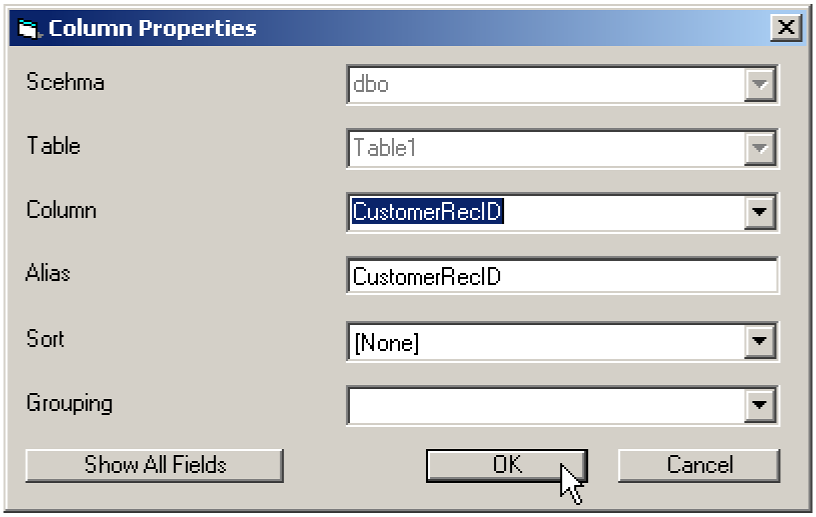
Control	Description / What to Do...
Schema	Select the appropriate schema in the `Schema` dropdown list.
Include Schema Name in Query	Select this check box if you want the schema name to be included as part of the query. This will add the schema specified in the query command. You can see the change in the query of `Database Interface Properties.` For example: `schema.<tablename>` instead of `<tablename>`Adding the schema name to the query allows the same `<tablename>` to be accessed under different schemas.
Table	The `Table` dropdown list contains a list of the tables contained in the selected database. *Table names cannot contain spaces. Table names with spaces will not appear in the dropdown list.*
Key Column	(Required for Update queries) You must tell Genesys Agent Scripting which column in the table can be used to uniquely identify rows to update by selecting a column from this dropdown list. *Column names cannot contain spaces. Column names with spaces will not appear in the dropdown list.*
OK	Click `OK` when table properties have been set. Agent Scripting returns you to the `Database Interface Properties` dialog box where you define more details about your query.

End of procedure

Procedure: Add Column

Start of procedure

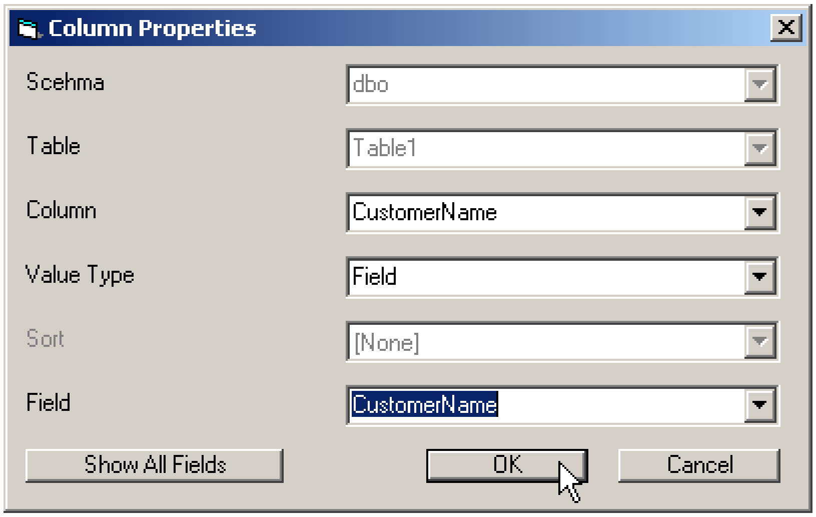
1. Select the Table in the `Query` tab and click `Add Column` to open the `Column Properties` dialog box.



Scehma: dbo
Table: Table1
Column: CustomerRecID
Alias: CustomerRecID
Sort: [None]
Grouping:

Figure 77: Column Properties Dialog Box for a Select Query

The `Column Properties` dialog box is where you select the columns in the database table that your query is focused upon. The dialog box in Figure 77 shows the structure of `Column Properties` for a Select query. Figure 78 shows the structure of `Column Properties` for an Update query.



Scehma: dbo
Table: Table1
Column: CustomerName
Value Type: Field
Sort: [None]
Field: CustomerName

Figure 78: Column Properties Dialog Box for an Update Query

Table 37 describes the options in the `Column Properties` dialog box.

Table 37: Column Properties

Control	Description / What to Do...
Schema	Displays the previously-selected schema for the database.
Table	Displays the previously-selected database table.
Column	The `Column` dropdown list contains a list of the columns contained in the selected database table. Select a column that you want to operate on. *Column names cannot contain spaces. Column names with spaces will not appear in the dropdown list.*
Alias	(Available for Select queries only) This field allows you to assign an alias for the data in the selected column. This can be useful when the database column name is very large, in which case you may want to assign an alias with a shorter name. The `Result` tab will then display the alias name of the column instead of the actual column name. By default, this field is populated with the value in the `Column` field.
Value Type	(Available for Update queries only) If you select `Field` from the dropdown list, the value used to update the database column comes from an Agent Scripting Field which you select in `Field` below. If you select `Value`, from the dropdown list, the value used to update the database column is a static value that you type into the `Value` field below.
Sort	(Available for Select queries only) Select `[None], Ascending,` or `Descending` to sort the retrieved data from the database column.

Table 37: Column Properties (Continued)

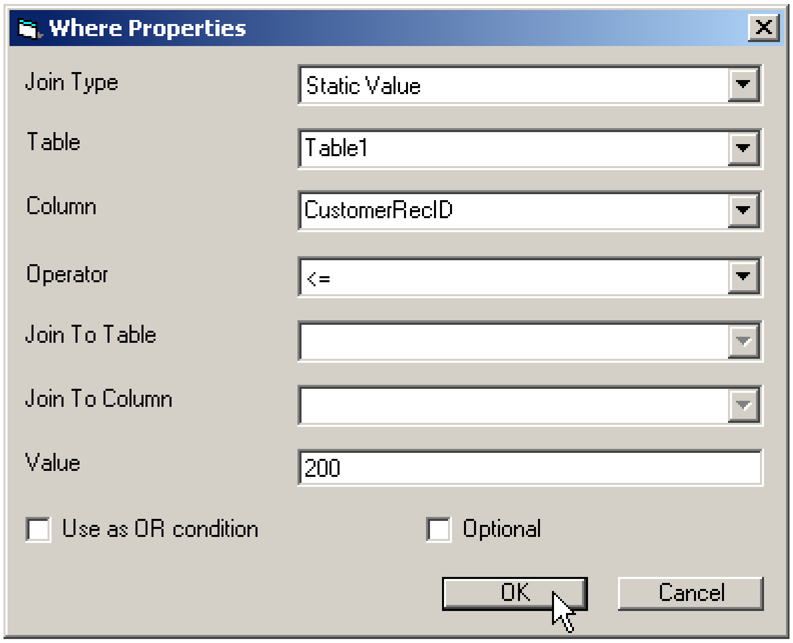
Control	Description / What to Do...
Grouping	(Available for Select queries only) This control allows you to group the column in an SQL query in order to perform one of the following operation: `Max` = Return the maximum value `Min` = Return the minimum value `Count` = Return the number of values `Sum` = Return the sum of all values. In the `Result` tab you can map the result of the column operation to a Field.
Value	(Available for Update queries only) If the `Value Type` is `Value`, type the static value that is to be used to update this database column.
Field	(Available for Update queries only) If the `Value Type` is `Field`, select the Agent Scripting Field that contains the data to be used to update this database column.
Show All Fields	This button only applies for Update from Table queries. For queries of this type, you select a Genesys Agent Scripting table to map back to the database to be updated. When you then add a column to the query, the list of columns only contains those choices that are in the Genesys Agent Scripting table. The `Show All Fields` button will change the dropdown list so that it contains all Genesys Agent Scripting Fields rather than just those contained in the selected table.
OK	Click `OK` when column properties have been set. Agent Scripting returns you to the `Database Interface Properties` dialog box where you can define more details about your query.

End of procedure

Procedure: Add Where

Start of procedure

1. Select the Table in the `Query` tab and click `Add Where` to open the `Where Properties` dialog box as shown in Figure 79.



Where Properties

Join Type: Static Value

Table: Table1

Column: CustomerRecID

Operator: <=

Join To Table:

Join To Column:

Value: 200

□ Use as OR condition □ Optional

Figure 79: Where Properties Dialog Box

The `Where Properties` dialog box is where you place conditions on the query. You can have from zero to many Where clauses for a query.

Table 38 describes the options in the `Where Properties` dialog box.

Table 38: Where Properties

Control	Description / What to Do...
Join Type	The Join Type indicates how the rows in the query will be joined together. The options you can select from the dropdown list are: Static Value = You will provide a static value in the Value field which is used to select rows from the indicated table. Field = The value from a Genesys Agent Scripting Field will be used to select rows from the indicated table. Join = The values from another table and column in the selected database will be used to select rows from the table. Parent Join = Only valid in a nested query, this option allows you to join rows in this table with a column value from a higher-level query.
Table	Select the desired database table from the Table dropdown list. This table will be joined in the query.
Column	Select a column within the table from the Column dropdown list.
Operator	Select the type of join in the Operator dropdown list. Operators include =, >, <, and so on.
Join To Table	Only valid if the Join Type is set to Join, you select another table from the dropdown list of tables in the database that is part of this query.
Join To Column	Only valid if the Join Type is set to Join, you select a column within the Join To Table or parent column in a nested join from the dropdown list.
Value	Type a static value or select a Genesys Agent Scripting Field.

Table 38: Where Properties (Continued)

Control	Description / What to Do...

Use as OR condition	Select this check box to add an OR condition to the database query. Conditions will be grouped together based on the order in the tree. The OR condition applies to the WHERE condition that is selected *and the one immediately preceding it*.
	As an example, for the following conditions:
	WHERE A = B
	WHERE C = D (and Use as OR condition is selected)
	WHERE E = F
	The result will be:
	WHERE (A = B OR C = D) AND E = F

Optional	If a Where clause is marked as Optional, it is dropped from the query if the value of the Field is an empty string.
OK	Click OK when Where properties have been set. Agent Scripting returns you to the Database Interface Properties dialog box where you define more details about your query.

End of procedure

Mapping Results

The Query tab of Database Interface Properties defined the information that will be retrieved from the database during the query. Mapping results tells Agent Scripting where the data will go once it is retrieved from the database; that is, how to map the retrieved data to previously-defined Fields.

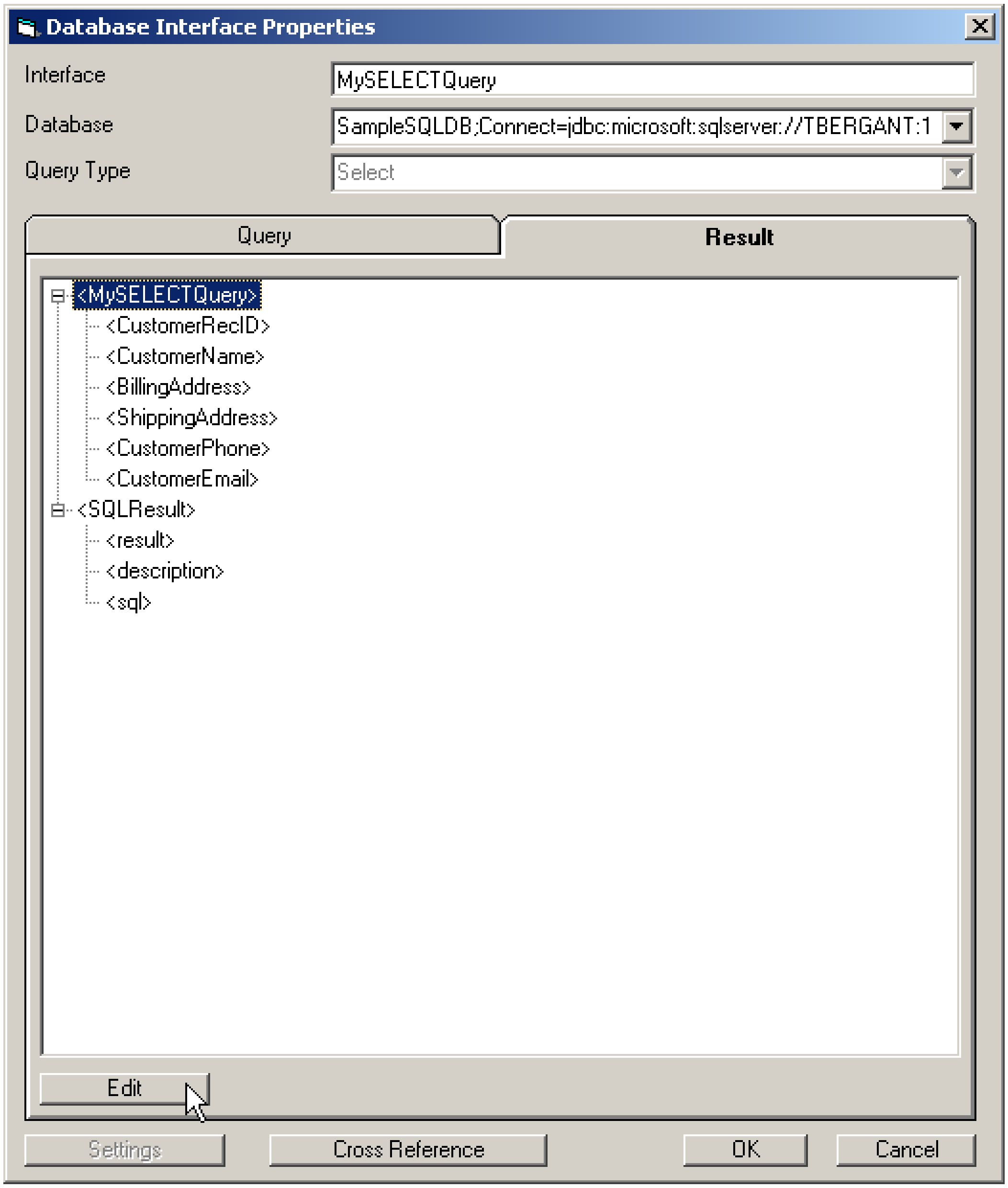
Genesys Agent Scripting offers two methods of mapping query results to Agent Scripting Fields:

- Using the Result tab
- Using the Map button

Procedure: Using the Result Tab

Start of procedure

1. Click `Result` to open the `Result` tab in the `Database Interface Properties` dialog box as shown in Figure 80 on page 193.



Database Interface Properties

Interface: MySELECTQuery

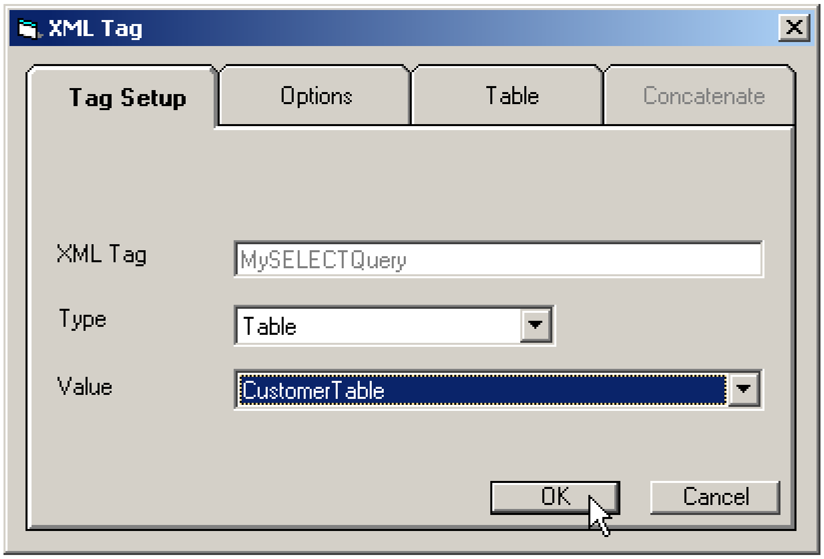
Database: SampleSQLDB;Connect=jdbc:microsoft:sqlserver://TBERGANT:1

Query Type: Select

Figure 80: Database Interface Properties Dialog Box - Result Tab

The database table columns you defined in your query of type Select appear in the Result field below the query name. You can edit the XML tags to map these database table columns directly to a Table, String, or Numeric field in Agent Scripting.

2. Select the query name or a database table column and click Edit to open the XML Tag dialog box for the query element as shown in Figure 81 on page 194.



XML Tag: MySELECTQuery

Type: Table

Value: CustomerTable

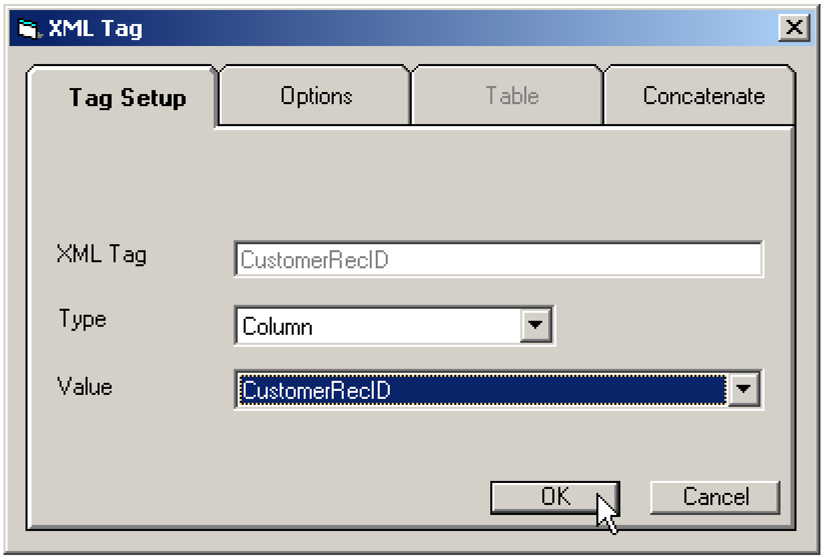
Figure 81: XML Tag Dialog Box for MySELECTQuery

The XML Tag dialog box is where you define the mapping between the external database and the Agent Scripting Field(s) that will hold the retrieved data.

Table 39 describes the options on the Tag Setup tab of the XML Tag dialog box.

Table 39: XML Tag Dialog Box - Tag Setup

Control	Description / What to Do...
XML Tag	Displays the previously-provided query name.
Type	Select the type of query element that applies to what you selected in the Result field. Select Table if you are setting up results for the query name (MySELECTQuery, for example, as shown in Figure 81). Select Column if you are setting up results for a database table column (CustomerRecID, for example, as shown in Figure 82 on page 196). Select Field if you want data to be stored in a Field.
Value	Select the Table field, or associated Field within a Table field, that will hold the data. If the Type is set to Table, existing Table fields will display in the Value dropdown list. If the Type is set to Column, Fields associated with the selected Table field will display in the Value dropdown list. If the Type is set to Field, possible Fields to store the data will display in the Value dropdown list.
OK	Click OK when the mapping has been set for all database columns. Agent Scripting returns you to the Database Interface Properties dialog box.



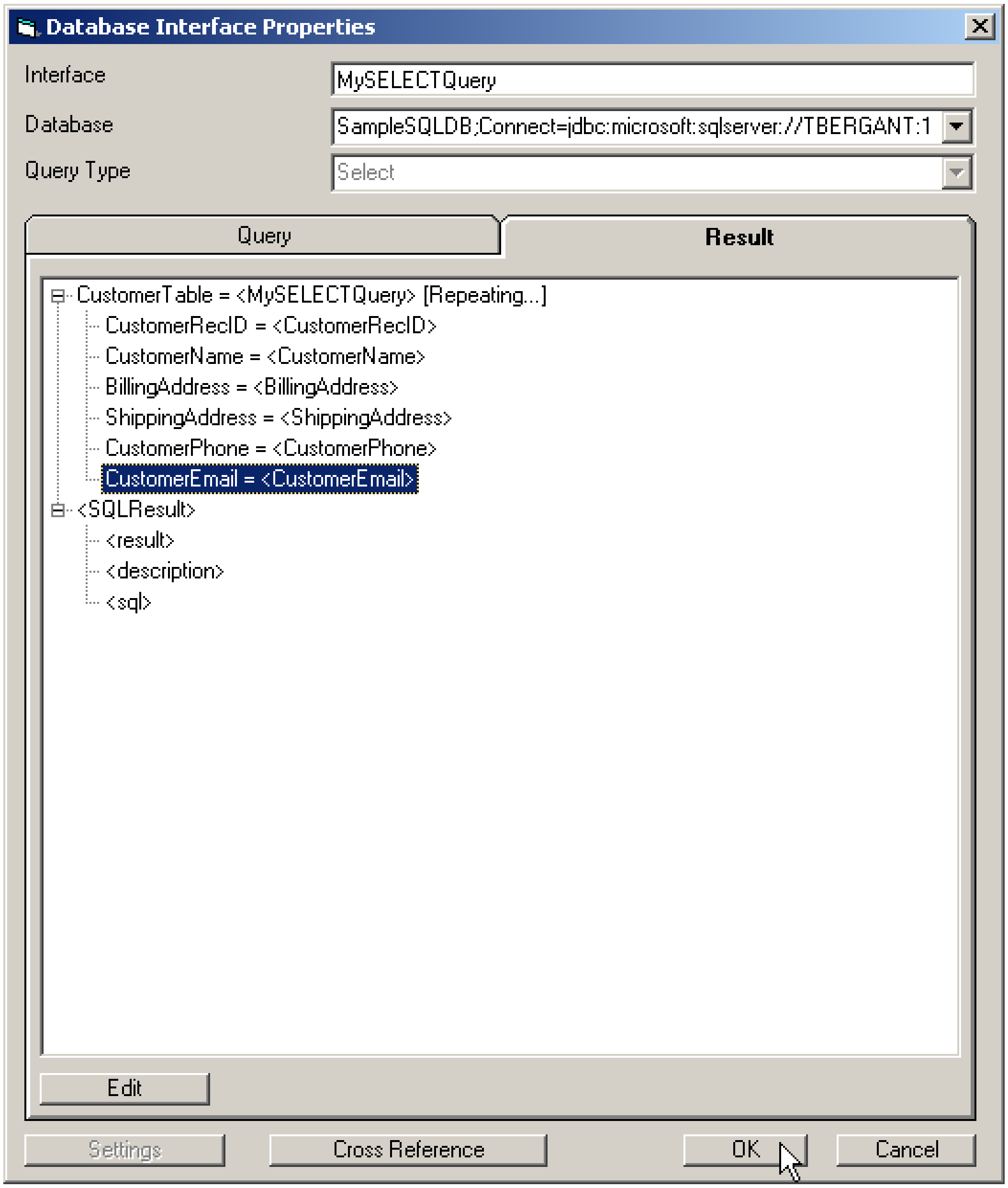
XML Tag CustomerRecID

Type Column

Value CustomerRecID

Figure 82: XML Tag Dialog Box for Column Type

Once you have defined Result mappings, the `Result` tab in the `Database Interface Properties` dialog box shows these mappings. See Figure 83 on page 197 for an example.



Interface: MySELECTQuery

Database: SampleSQLDB;Connect=jdbc:microsoft:sqlserver://TBERGANT:1

Query Type: Select

```
⊟ CustomerTable = <MySELECTQuery> [Repeating...]
    CustomerRecID = <CustomerRecID>
    CustomerName = <CustomerName>
    BillingAddress = <BillingAddress>
    ShippingAddress = <ShippingAddress>
    CustomerPhone = <CustomerPhone>
    CustomerEmail = <CustomerEmail>
⊟ <SQLResult>
    <result>
    <description>
    <sql>
```

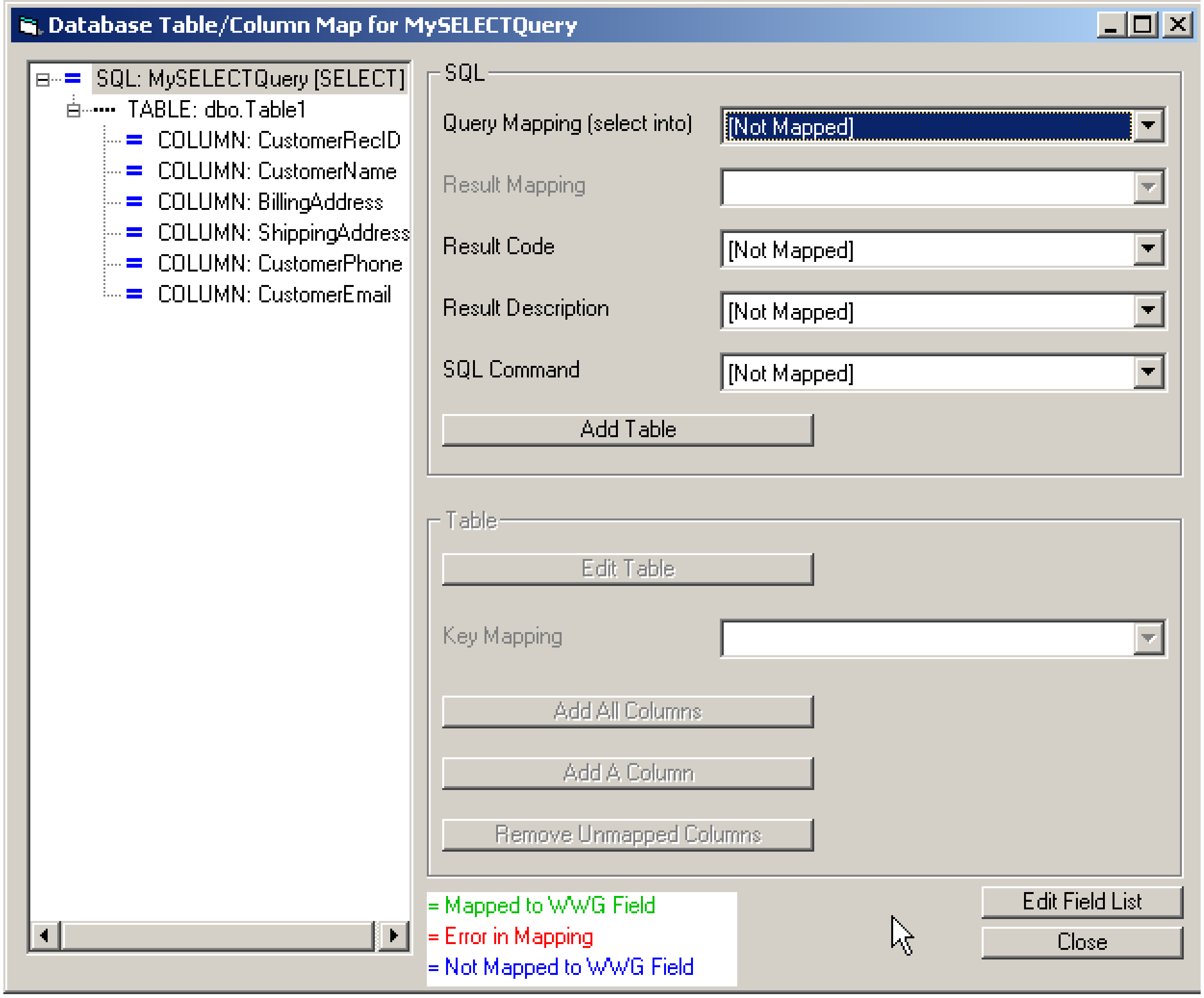
Figure 83: Database Interface Properties Dialog Box - Result Mappings

End of procedure

Procedure: Using the Map Button

Start of procedure

1. Click `Map` in the `Database Interface Properties` dialog box to open the `Database Table/Column Map` dialog box as shown in Figure 84 on page 198.



Database Table/Column Map for MySELECTQuery

- SQL: MySELECTQuery [SELECT]
  - TABLE: dbo.Table1
    - COLUMN: CustomerRecID
    - COLUMN: CustomerName
    - COLUMN: BillingAddress
    - COLUMN: ShippingAddress
    - COLUMN: CustomerPhone
    - COLUMN: CustomerEmail

SQL

Query Mapping (select into): [Not Mapped]

Result Mapping:

Result Code: [Not Mapped]

Result Description: [Not Mapped]

SQL Command: [Not Mapped]

Table

Key Mapping:

= Mapped to WWG Field

= Error in Mapping

= Not Mapped to WWG Field

Figure 84: Database Table/Column Map Dialog Box

This dialog box is where you define the mapping between the external database and the Agent Scripting Field(s) that will hold the retrieved data.

Table 40 on page 199 describes the options on the `Database Table/Column Map` dialog box.

Table 40: Database Table/Column Map Dialog Box

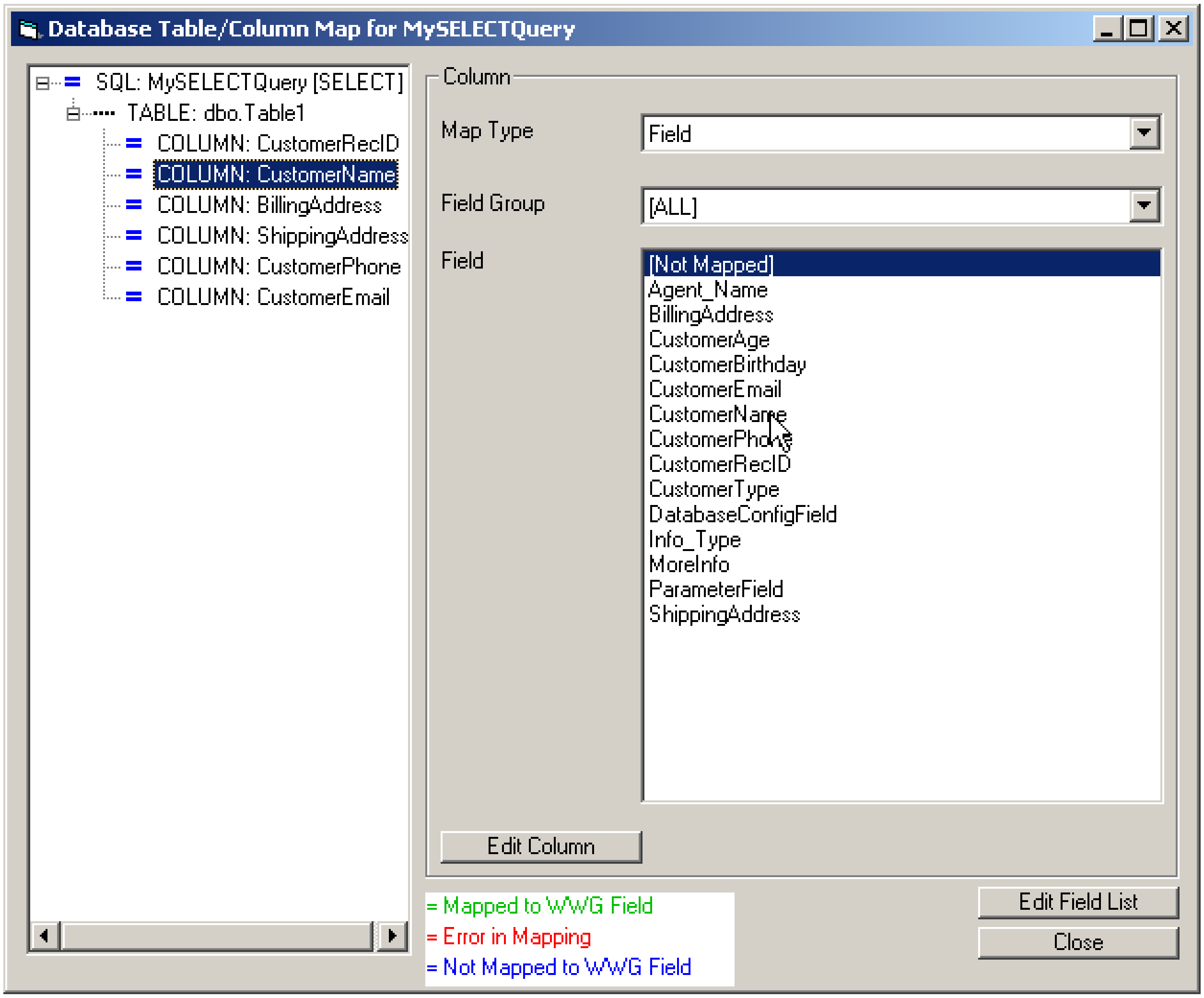
Control	Description / What to Do...
SQL	Contains dropdown lists pertaining to query and result mapping. These options are only available when a Query or Table is selected in the left pane of the dialog box.
Query Mapping (select info)	(Available for Select queries only.) Select the Table Field from the dropdown list to which you will map the SQL query results (values returned by the SQL query). This is useful if the query is returning more than one record from the database.
Query Mapping (update from)	(Available for Update queries only.) Select `Insert, Update, Delete,` or a predefined Agent Scripting Table Field to indicate where the database values are being updated from.
Result Mapping	(Available for Update queries only.) `Result Mapping` is used on Update queries for returned values. An Update query might return multiple values if inserting from an Agent Scripting table. Each insert will generate a key. You can set the result mapping to a table. This will cause the key mapping in the table section to just show columns within that table.
Result Code	`Result Code` allows you to map the numeric result code returned from the SQL query to a Genesys Agent Scripting Field. In order to do so, select the Field to which to map the result code from the dropdown list.
Result Description	`Result Description` allows you to map the result description returned from the SQL query to an Agent Scripting Field. In order to do so, select the Field to which to map the result description from the dropdown list.
SQL Command	`SQL Command` allows you to map the SQL query that was executed to an Agent Scripting Field. In order to do so, select the Field to which to map the SQL command from the dropdown list.

Table 40: Database Table/Column Map Dialog Box (Continued)

Control	Description / What to Do...
Add Table	Click `Add Table` to add a table to the query as described in “Add Table” on page 185.
Table	Contains dropdown lists pertaining to query and result mapping. These options are only available when a Table is selected in the left pane of the dialog box.
Edit Table	`Edit Table` opens the `Table Properties` dialog box and allows you to edit information for the selected table.
Key Mapping	(Available for Update queries only.) Each insert into the database generates a key. `Key Mapping` allows you to map the contents of an Agent Scripting field as the key. The dropdown list allows you to select which Field should generate the key. If the Update query is inserting from an Agent Scripting table, the `Key Mapping` field will only allow you to select from columns within that table.
Add All Columns	Click `Add All Columns` to add all columns from the selected table to the query. This button opens the `Column Properties` dialog box. If a column has already been added to the table query, it will not add it a second time.
Add A Column	Click `Add A Column` to add a column to the selected table in the query as described in “Add Column” on page 187.
Remove Unmapped Columns	Click `Remove Unmapped Columns` to remove any unmapped columns from the query. Results will no longer be returned for those columns.

Table 40: Database Table/Column Map Dialog Box (Continued)

Control	Description / What to Do...
Column	The right pane of the Database Table/Column Map dialog box changes when a column is selected in the left pane, as shown in Figure 85 on page 202. The controls in the Column area are available only when a column is selected.
Map Type	Map Type tells you the type of mapping that occurs. Database columns can be mapped to Columns or Fields. If multiple rows are being returned and the results are being written to an Agent Scripting Table Field, the database columns should be mapped to columns in the Agent Scripting Field. If a single result is returned, database columns should be mapped to Fields.
Field Group	Field Group filters the Field List based on group. Select the Field Group from the dropdown list and the Field List will only display those Fields that belong to the selected group.
Field	The Field box lists all of the possible Agent Scripting Fields to which the selected column may be mapped. Select a Field from the list and the column selected in the left pane of the dialog box will be mapped to it. To unmap a column, select [Not Mapped].
Edit Column	Edit Column opens the Column Properties dialog box and allows you to edit information for the selected column.
Edit Field List	(Available at all times) Click Edit Field List to open the Field List dialog box. From this dialog box, you can add, edit, or delete Fields.
Close	(Available at all times) Click Close to close the dialog box. Agent Scripting returns you to the Database Interface Properties dialog box.



Database Table/Column Map for MySELECTQuery

SQL: MySELECTQuery [SELECT]
TABLE: dbo.Table1
COLUMN: CustomerRecID
COLUMN: CustomerName
COLUMN: BillingAddress
COLUMN: ShippingAddress
COLUMN: CustomerPhone
COLUMN: CustomerEmail

Column

Map Type: Field

Field Group: [ALL]

Field:
[Not Mapped]
Agent_Name
BillingAddress
CustomerAge
CustomerBirthday
CustomerEmail
CustomerName
CustomerPhone
CustomerRecID
CustomerType
DatabaseConfigField
Info_Type
MoreInfo
ParameterField
ShippingAddress

= Mapped to WWG Field
= Error in Mapping
= Not Mapped to WWG Field

Figure 85: Database Table/Column Map Dialog Box - COLUMN Selected

Completing the Database Interface

2. After you have mapped all database table columns to the Fields associated with the Agent Scripting Table Field, click `OK` in the `Database Interface Properties` dialog box to complete the Database Interface.

The interface (query) that you defined appears in the Database Interface List.

End of procedure

Examples

This section shows three step-by-step examples that illustrate how to set up Database connections and Database Interfaces:

- Creating a Database Connection
- Setting Up a `SELECT` Database Interface to retrieve data from an external database table
- Setting Up an `UPDATE` Database Interface to insert data into an external database table

Creating a Database Connection

Let's assume you have data in an external SQLServer database called `SampleSQLDB`, and that you want a compiled JSP Agent Scripting application to have access to the data. You should create a Database connection.

Procedure: Create a Database Connection

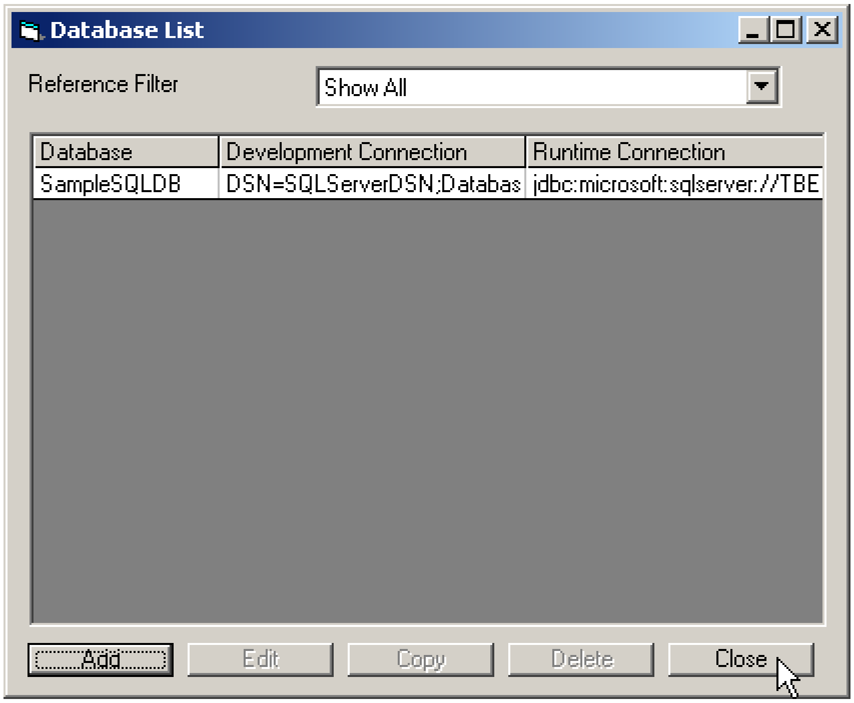
Start of procedure

1. Select `Define` > `Databases`.
2. Click `Add` in the `Database List` dialog box.
3. In the `Database Properties` dialog box, type a name for this Database connection in the `Database Name` field.

   This does not have to be the same as your actual external database name.
4. Select `SQLServer` from the `Database Type` dropdown list.
5. Find or create a Data Source Name (DSN) for this database type in Windows under `Control Panel` > `Administrative Tools` > `Data Sources (ODBC)`.
6. Type your development connection string in the `Development Connection` field. An example might be:
7. `DSN=SQLServerDSN;Database=SampleSQLDB;UID=johnsmythe;PWD=astro22`
8. Select `Static` in the `Provider` dropdown list, then type or select `com.microsoft..sqlserver.jdbc.SQLServerDriver` in the adjacent combo box.
9. Select `Static` in the `Runtime Connection` dropdown list, then type your runtime connection string in the adjacent field. An example might be:

   ```
   jdbc:microsoft:sqlserver://MyMachine:1433;Database=SampleSQLDB;User=johnsmythe;Password=astro22
   ```
10. Click `OK`.

    Your new Database connection is added to the `Database List` as shown in Figure 86 on page 204.



Reference Filter: Show All

Database	Development Connection	Runtime Connection
SampleSQLDB	DSN=SQLServerDSN;Databas	jdbc:microsoft:sqlserver://TBE

Figure 86: Database List - Database Connection Added

End of procedure

Setting Up a SELECT Database Interface

Let’s create a Database Interface that selects data from an external database table and delivers the data to a defined Agent Scripting Table Field.

Note: This example requires an external database and a `Table` field that has been defined with customer information `String` fields. See the chapter called “Creating Fields”for details on how to create `String` and `Table` fields.

Procedure:
Set Up a SELECT Database Interface

Start of procedure

1. Select `Define` > `Database Interfaces`.
2. Click `Add` in the `Database Interface List` dialog box.
3. In the `Database Interface Properties` dialog box, type a name for this interface in the `Interface` field.

4. Select the Database connection from the Database dropdown list.
5. Select Select from the Query Type dropdown list.
6. Click Add Query to open the Query Properties dialog box.
7. Type a query name in the Query Name field.
8. Click OK to return to the Database Interface Properties dialog box.

   The SELECT query is added to the Query field.
9. Select the query line in the Query field and click Add Table to open the Table Properties dialog box.
10. Select the database schema from the Schema dropdown list.
11. Select the database table for your query from the Table dropdown list.
12. Click OK to return to the Database Interface Properties dialog box.

    The table query element is added to the Query field below the query name.
13. Select the table line in the Query field and click Add Column to open the Column Properties dialog box.
14. Select a database table column for your query from the Column dropdown list.

    The Alias field is populated with the same information. You may change the alias if you wish.
15. Click OK to return to the Database Interface Properties dialog box.

    The column query element is added to the Query field below the table query element.

    Let's add more columns to the query using the Map operation.
16. Click the Map button to open the Database Table/Column Map dialog box.
17. Select the Table element in the left pane of the dialog box.
18. Click Add A Column and follow Step 14 above when you see the Column Properties dialog box. Click OK to add the selected column.
19. Repeat Step 18 until you've added all desired columns to the query.
20. Select the first Column element in the left pane of the dialog box.
21. In the Field box, click the Agent Scripting Field name that you want to map to the selected column.

    The column and Field are mapped as soon as you click the Field name.
22. Select another Column element in the left pane and repeat Step 21.
23. Map all remaining columns to Agent Scripting Fields in the same manner.
24. Click Close in the Database Table/Column Map dialog box to return to the Database Interface Properties dialog box.
25. Click OK in the Database Interface Properties dialog box to finish defining the Select query Database Interface.

Your new Database Interface is added to the Database Interface List.

End of procedure

Setting Up an UPDATE Database Interface

Let's create a Database Interface that inserts data from an Agent Scripting Table field into an external database table.

Note: This example requires an external database and a Table field that has been defined with customer information String fields. See the chapter called "Creating Fields"for details on how to create String and Table fields.

Procedure: Setting Up an UPDATE Database Interface

Start of procedure

1. Select Define > Database Interfaces.
2. Click Add in the Database Interface List dialog box.
3. In the Database Interface Properties dialog box, type a name for this interface in the Interface field.
4. Select the Database connection from the Database dropdown list.
5. Select Update from the Query Type dropdown list.
6. Click Add Query to open the Query Properties dialog box.
7. Type a query name in the Query Name field.
8. Select Insert from the Update Type dropdown list.
9. Click OK to return to the Database Interface Properties dialog box.

   The UPDATE Insert query is added to the Query field.
10. Select the query line in the Query field and click Add Table to open the Table Properties dialog box.
11. Select the database schema from the Schema dropdown list.
12. Select the database table for your query from the Table dropdown list.
13. Select the database's key column from the Key Column dropdown list.
14. Click OK to return to the Database Interface Properties dialog box.

    The table query element is added to the Query field below the query name.
15. Select the table line in the Query field and click Add Column to open the Column Properties dialog box.

16. Select a database table column to be updated from the Column dropdown list.

17. Select Field from the Value Type dropdown list to indicate that the database column will be updated with the data in an Agent Scripting Field.

18. In Field, select the Agent Scripting Field that contains the data which will update your chosen database column.

19. Click OK to return to the Database Interface Properties dialog box.

    The column query element is added to the Query field below the table query element.

20. Repeat Steps 15 through 19 for all other database table columns that you wish to add to this Update query.

21. After all columns and their mappings are added, click OK in the Database Interface Properties dialog box to finish defining the Update query Database Interface.

    Your new Database Interface is added to the Database Interface List.

End of procedure

Chapter

8 API Interfaces

This chapter provides an overview of how to create and use an API Interface.

The information in this chapter is divided among the following topics:

What Is an API Interface?

An API Interface enables Genesys Agent Scripting to execute custom code in a script. The API Interface can provide information to the custom code, and it can also update defined fields with information that it receives for use in a script.

Creating an API Interface

You will use the `Define` menu.

Procedure:
Create an API Interface

Start of procedure

1. Select `Define` > `API Interfaces` to open the `API Interface List` dialog box (see Figure 87). From the `API Interface List` dialog box, you can edit, copy, delete, or add API Interfaces.

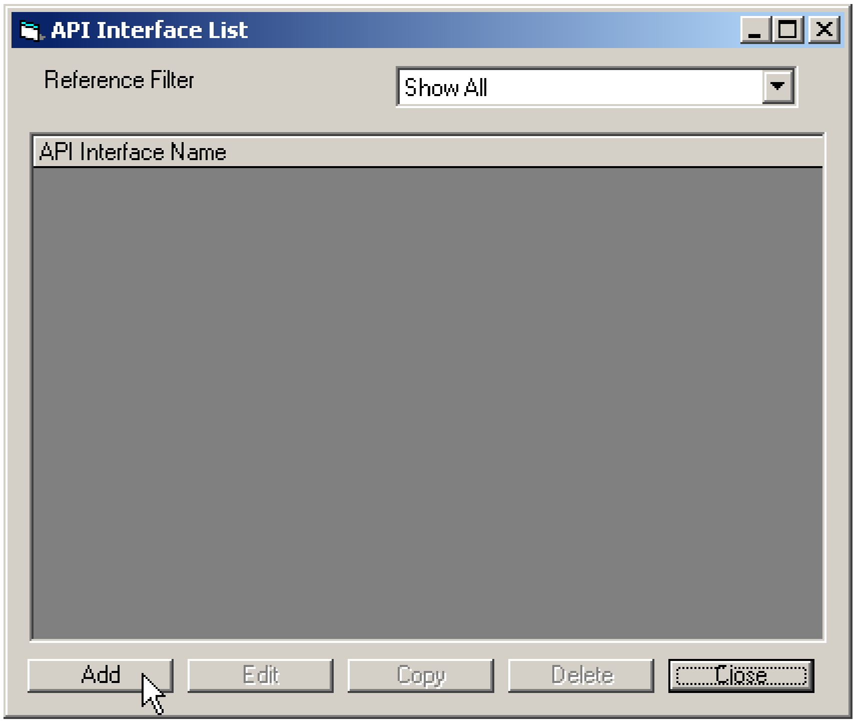


Figure 87: API Interface List Dialog Box

2. Click `Add` to open the `API Interface Properties` dialog box (see Figure 88).

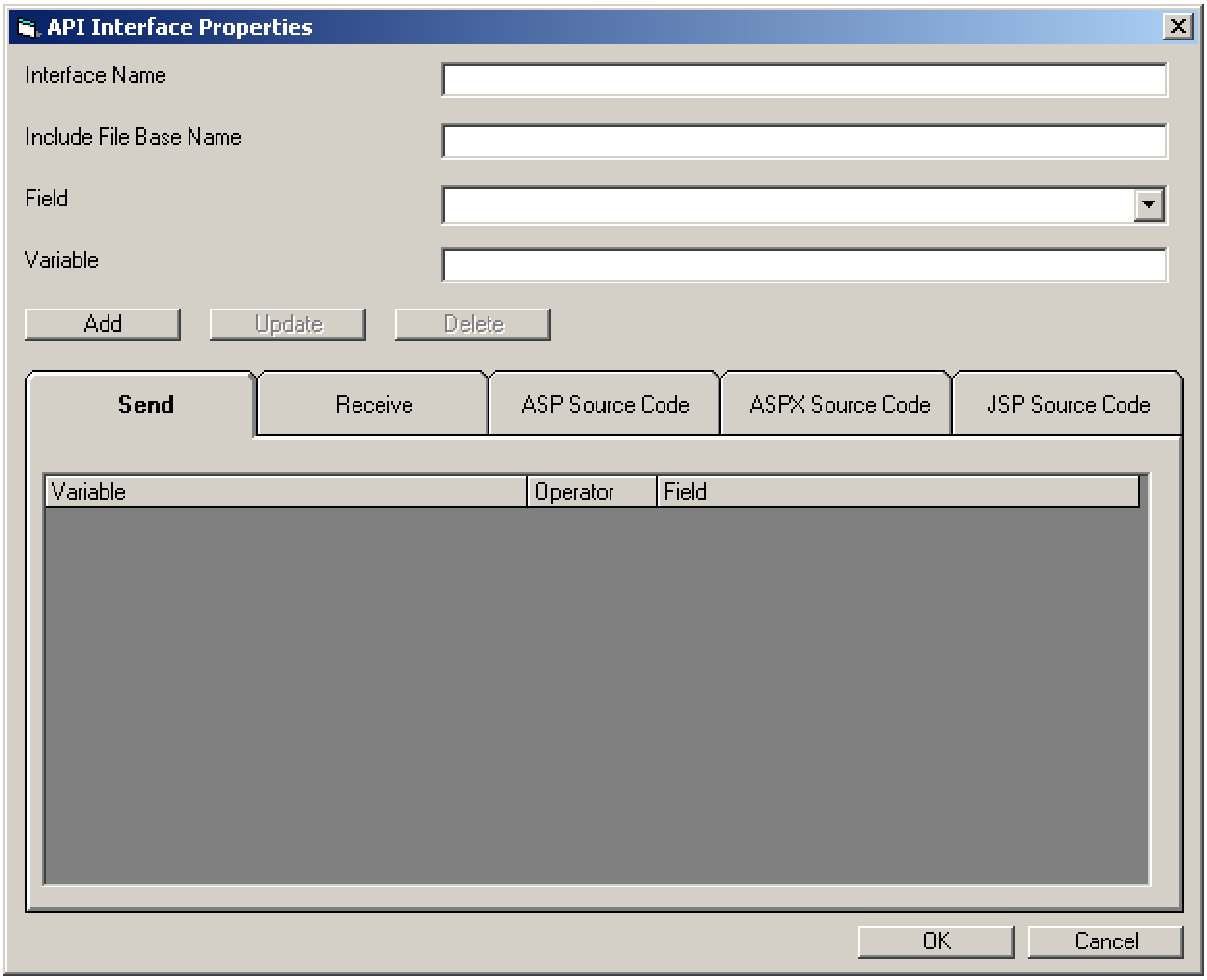


Figure 88: API Interface Properties Dialog Box

Table 41 describes the options on the API Interface Properties dialog box.

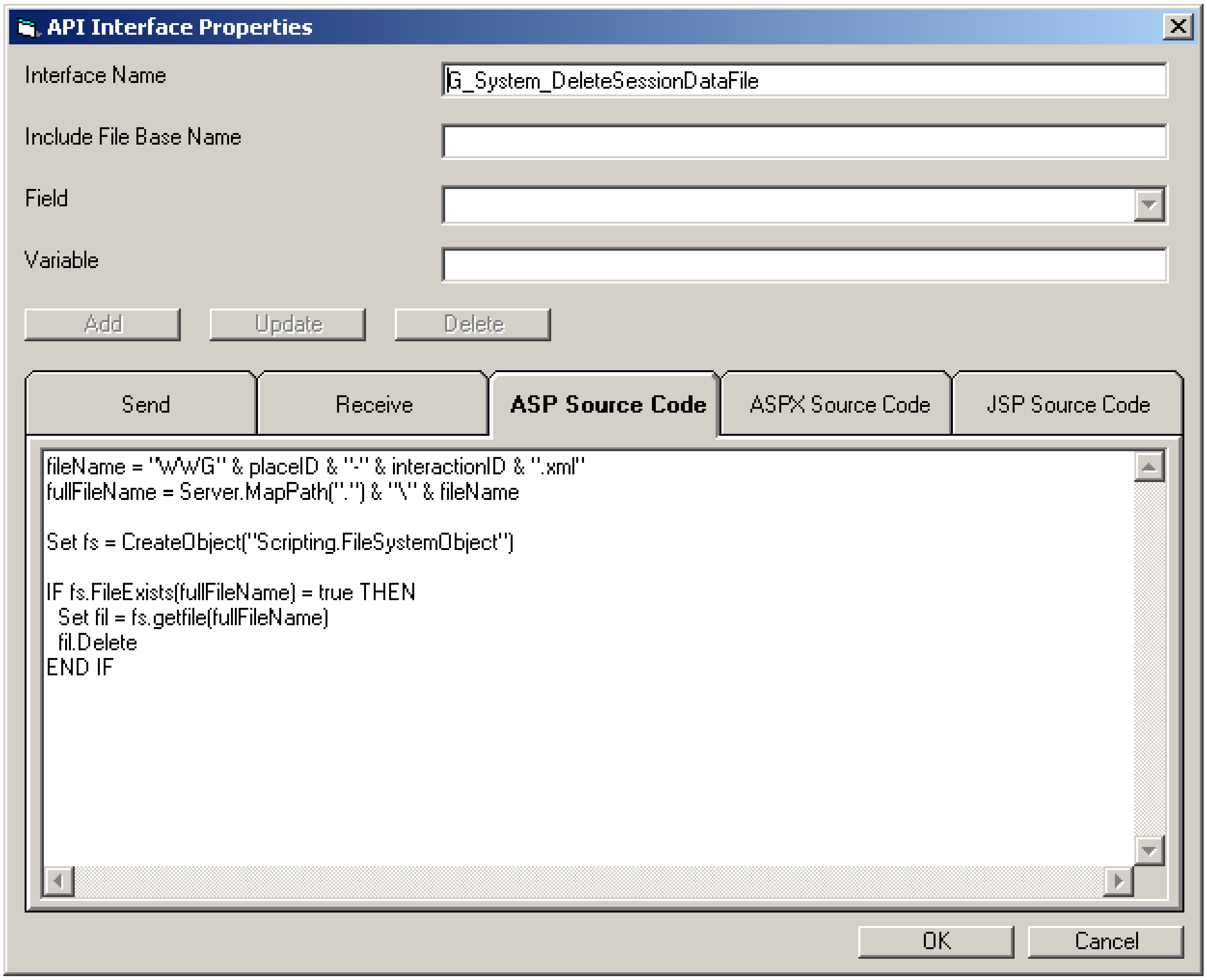
Table 41: API Interface Properties

Control	Description / What to Do...
Interface Name	Enter a name for the API Interface.
Include File Base Name	Enables you to add an Include file to a call code outside of Genesys Agent Scripting. Include files are files required in order to support the API function.
Field	Use the Field drop-down list to select a Genesys Agent Scripting field to associate to a variable. This field is active only if the Send tab or Receive tab is selected.

Table 41: API Interface Properties (Continued)

Control	Description / What to Do...
Variable	Enter a variable name associated with the selected field. The Variable field is enabled only if the Send tab or Receive tab is selected.
Add	Click Add to enter the Interface Name, Field, and Variable information onto the appropriate tab (Send or Receive).
Update	Click Update to apply changes made to the Interface Name, Field, and Variable information on the appropriate tab (Send or Receive).
Delete	Click Delete to delete selected rows from the Send or Receive tab.
Send Tab	Used to define variables that will receive values from Genesys Agent Scripting fields before the API function is executed. These variables must be defined in the called function.
Receive Tab	Used to define variables that will send data to Genesys Agent Scripting fields after the execution of the API function.
ASP Source Code Tab	Used to define the code that the API Interface will execute for ASP target environments. Write this code using Visual Basic. For an example, see "ASP Source Code Tab—Example" on page 213.
ASPX Source Code Tab	Used to define the code that the API Interface will execute for ASPX target environments. The code should be written in C#. For an example, see "ASPX Source Code Tab—Example" on page 214.
JSP Source Code Tab	Used to define the code that the API Interface will execute for JSP target environments. The code should be written in Java. For an example, see "JSP Source Code Tab—Example" on page 215.

ASP Source Code Tab—Example



Interface Name: G_System_DeleteSessionDataFile

Include File Base Name

Field

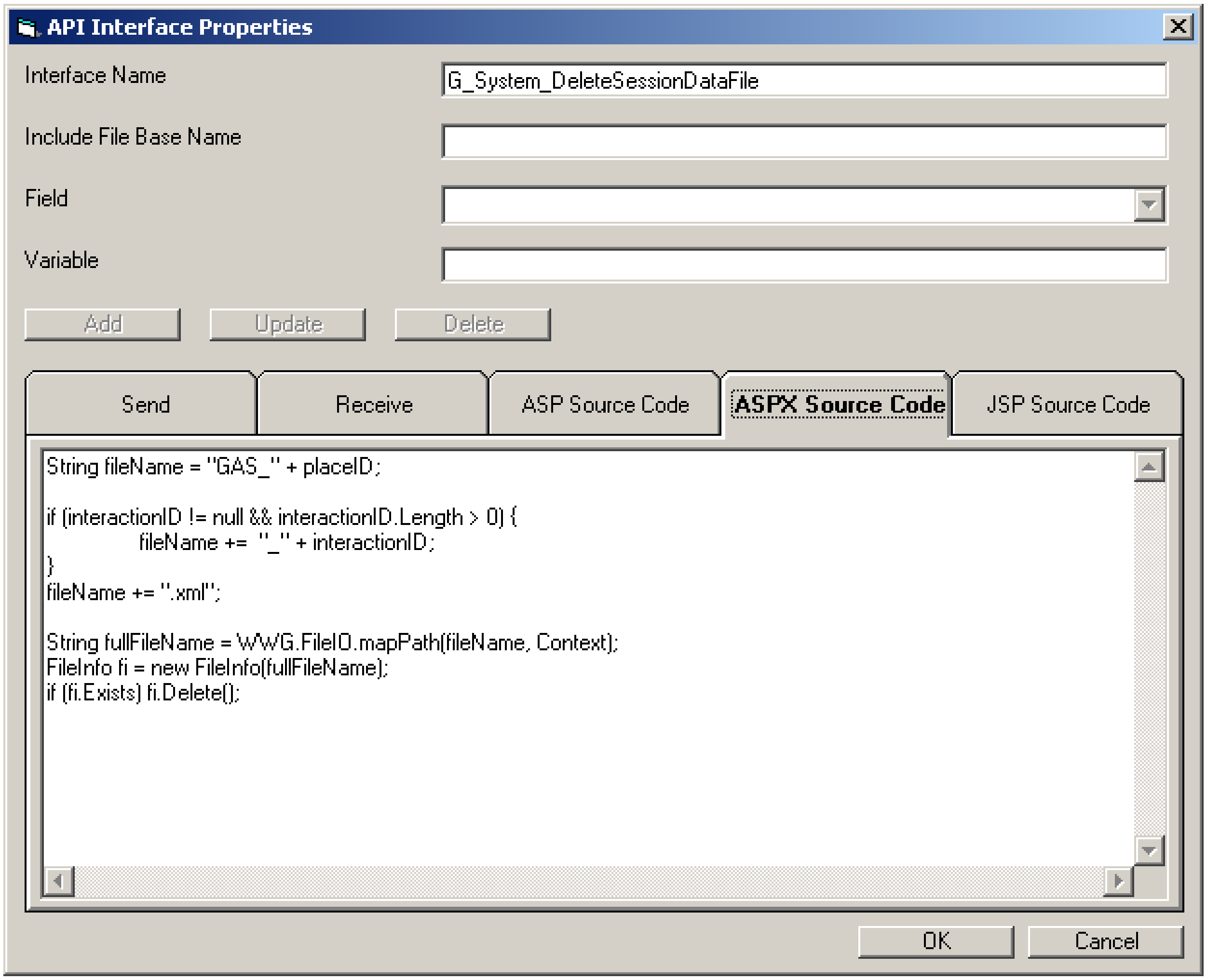
Variable

```
fileName = "WWG" & placeID & "-" & interactionID & ".xml"
fullFileName = Server.MapPath(".") & "\" & fileName

Set fs = CreateObject("Scripting.FileSystemObject")

IF fs.FileExists(fullFileName) = true THEN
  Set fil = fs.getfile(fullFileName)
  fil.Delete
END IF
```

Figure 89: API Interface Properties—ASP Source Code Tab

The custom code shown in the `ASP Source Code` tab is written using Visual Basic.

ASPX Source Code Tab—Example



Interface Name: G_System_DeleteSessionDataFile

Include File Base Name:

Field:

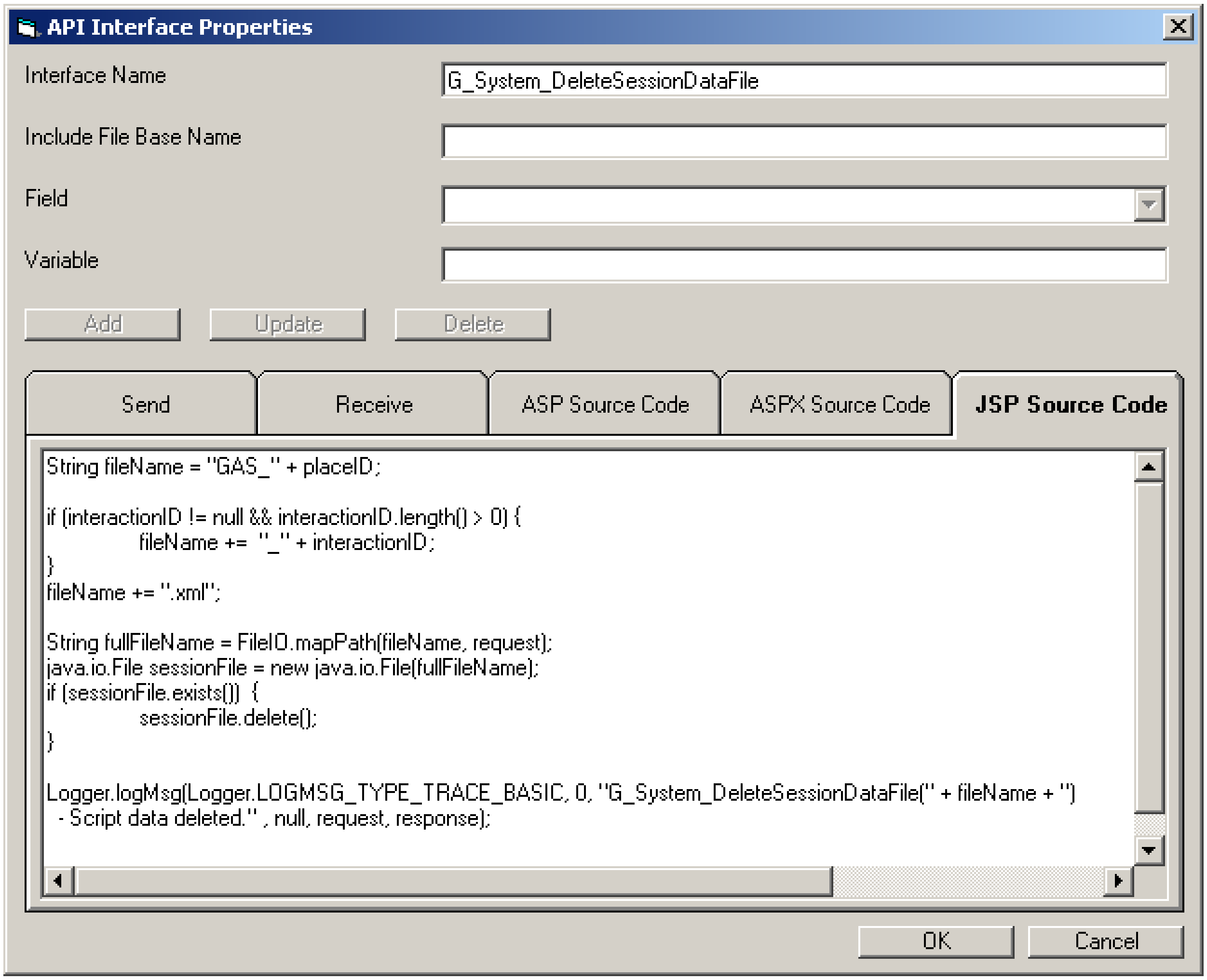
Variable:

```
String fileName = "GAS_" + placeID;

if (interactionID != null && interactionID.Length > 0) {
		fileName +=  "_" + interactionID;
}
fileName += ".xml";

String fullFileName = WWG.FileIO.mapPath(fileName, Context);
FileInfo fi = new FileInfo(fullFileName);
if (fi.Exists) fi.Delete();
```

Figure 90: API Interface Properties—ASPX Source Code Tab

The custom code shown in the `ASPX Source Code` tab is written using C#.

JSP Source Code Tab—Example



Interface Name: G_System_DeleteSessionDataFile

Include File Base Name:

Field:

Variable:

```
String fileName = "GAS_" + placeID;

if (interactionID != null && interactionID.length() > 0) {
            fileName +=  "_" + interactionID;
}
fileName += ".xml";

String fullFileName = FileIO.mapPath(fileName, request);
java.io.File sessionFile = new java.io.File(fullFileName);
if (sessionFile.exists())  {
            sessionFile.delete();
}

Logger.logMsg(Logger.LOGMSG_TYPE_TRACE_BASIC, 0, "G_System_DeleteSessionDataFile(" + fileName + ")
 - Script data deleted." , null, request, response);
```

Figure 91: API Interface Properties—JSP Source Code Tab

The custom code shown in the `JSP Source Code` tab is written using Java.

3. Click OK to save your changes and Cancel to abandon them. Both selections close the dialog box.

End of procedure

Example

This section shows a step-by-step example that illustrates how to set up an API Interface. In this example, the custom Visual Basic code is checking for the existence of a file, and it will return a Boolean value.

Custom Code Example

On the ASP Source Code tab of the API Interface Properties dialog box (see Figure 89 on page 213), is the following sample code that checks for the existence of a file:

```
sessionIDExists = "FALSE"
fileName = "WWG" & placeID + "-" & interactionID & ".xml"
fullFileName = Server.MapPath(".") & "\" & fileName

Set fs = CreateObject("Scripting.FileSystemObject")

IF fs.FileExists(fullFileName) = true THEN
   sessionIDExists = "TRUE"
END IF
```

The Fields Used in This Example

There are three fields of type String that will be used in this example: G_System_SessionDataExists, placeID and interactionID. These fields can be created using Define > Fields. For details on creating fields, refer to “Creating Fields” on page 65.

Procedure: Set Up the API Interface

Let’s create an API Interface that checks for the existence of a data file, based on the preceding sample code.

Start of procedure

1. From the Genesys Agent Scripting menu, select Define > API Interfaces.
2. In the API Interface List dialog box, click Add.
3. In the API Interface Properties dialog box, type a name for this interface in the Interface Name box.
   In this example, we are using the name G_System_SessionDataFileExists.
4. Click the Send tab.

On the Send tab, we will define two string variables that will receive values from two Genesys Agent Scripting fields before the API function is executed. These variables are defined in the preceding custom code, and they will be used to define the file name that we are looking for.

5. From the Field drop-down list, select your pre-defined field of type String.
   In this example, we are selecting a field named G_Config_Place.

6. In the Variable box, type your variable name.
   In this example, we are using the name placeID.
7. Click Add to add this expression to the Send tab.
8. From the Field drop-down list, select your pre-defined field of type String.
   In this example, we are selecting a field named G_AIL_ID.
9. In the Variable box, type your variable name.
   In this example, we are using the name interactionID.
10. Click Add to add this expression to the Send tab.
11. Click the Receive tab.

On the Receive tab, we will define a variable that will be used to receive a Boolean value after the custom code has been executed.

12. From the Field drop-down list, select your pre-defined field of type String.
    In this example, we are selecting a field named G_System_SessionDataExists.
13. In the Variable box, type your variable name.
    In this example, we are using the name sessionIDExists.
14. Click Add to add this expression to the Receive tab.
15. Click OK.

Your new API Interface is added to the API Interface List.

End of procedure

Chapter

9 XML Interfaces

This chapter provides an overview of how to create and use an XML Interface.

The information in this chapter is divided among the following topics:

What Is an XML Interface?

An Extensible Markup Language (XML) Interface enables a script to post and receive information from other systems.

Note: In order to invoke an XML Interface, you need to create an Action. See "Creating Actions".

Creating an XML Interface

You will use the `Define` menu.

Procedure:
Create an XML Interface

Start of procedure

1. Select Define > XML Interfaces to open the XML Interface List dialog box (see Figure 92). From the XML Interface List dialog box, you can edit, copy, delete, or add XML Interfaces.

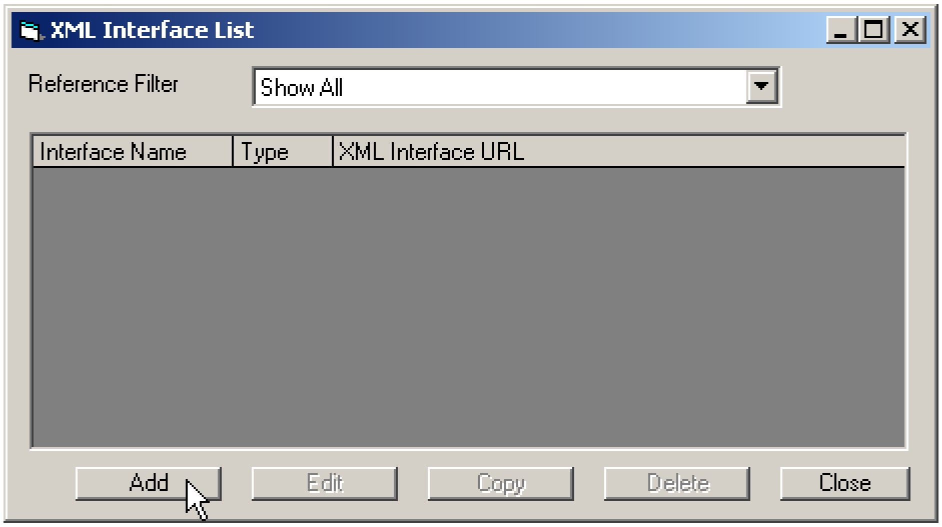
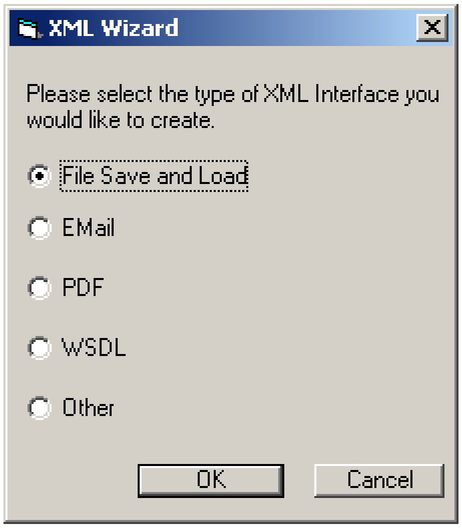


Figure 92: XML Interface List Dialog Box

2. Click Add to open the XML Wizard (see Figure 93).



Please select the type of XML Interface you would like to create.

● File Save and Load
○ EMail
○ PDF
○ WSDL
○ Other

Figure 93: XML Wizard

3. Select the radio button of an XML Interface type and click OK, or click Cancel to abandon this procedure.

See the section "XML Interface Types" below.

End of procedure

XML Interface Types

This section provides a brief description of the available XML Interface Types that you can select in the XML Wizard.

File Save and Load

Selecting the `File Save and Load` XML Interface type creates XML Interfaces to save fields and their contents to a flat file, and to load those fields and their contents from the same flat file into Genesys Agent Scripting. For more information, see the "Creating a File Save and Load XML Interface" section.

e-mail

Selecting the `e-mail` XML Interface type creates an XML Interface to enable e-mail to be sent from Genesys Agent Scripting. For more information, see "Creating an e-mail XML Interface" on page 226.

Note: If you added a table —for example, as the body for mail—you must also add, under the added table, an XML tag for every column in the table with an XML tag: `td`, type: `column`, Value: `field from table`.

In this case, received mail would include the specified table in table format.

WSDL

Selecting the `WSDL` XML Interface type creates XML Interfaces to enable a Web service definition to be automatically imported from a URL into an XML Interface. Genesys Agent Scripting currently uses Simple Object Access Protocol (SOAP) to connect to the Web service automatically. For more information, see "Creating a WSDL XML Interface" on page 230.

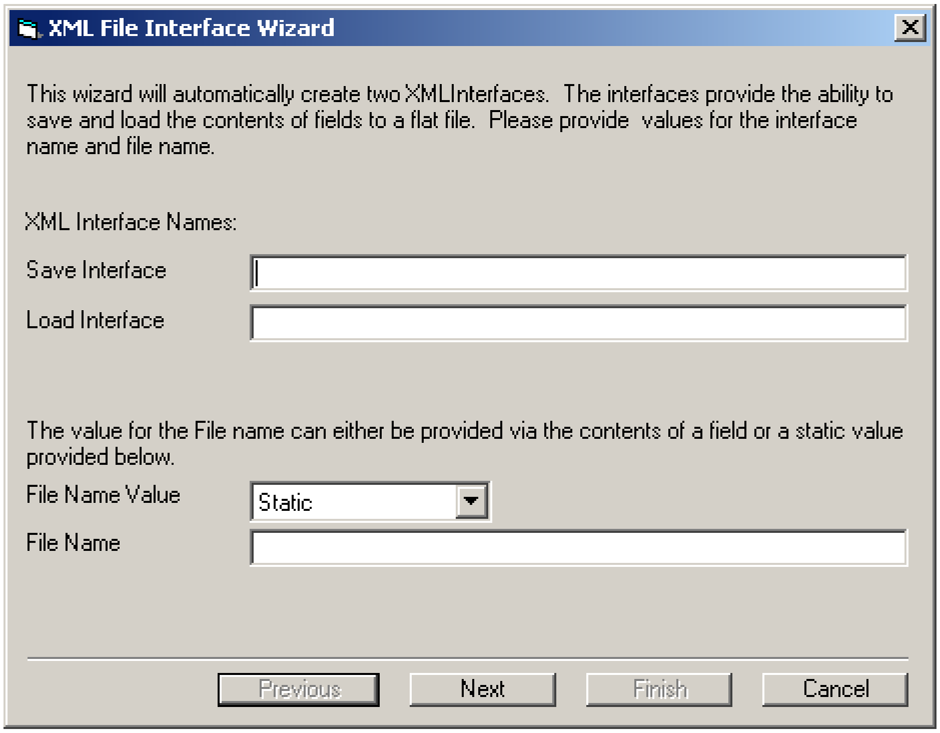
Other

Selecting `Other` enables you to create an XML Interface to communicate with other systems through XML. For more information, see "Creating an XML Interface—Other" on page 234.

Procedure: Creating a File Save and Load XML Interface

Start of procedure

1. From the XML Wizard (see Figure 93), select File Save and Load to open the first screen of the XML File Interface Wizard (see Figure 94).



XML File Interface Wizard

This wizard will automatically create two XMLInterfaces. The interfaces provide the ability to save and load the contents of fields to a flat file. Please provide values for the interface name and file name.

XML Interface Names:

Save Interface

Load Interface

The value for the File name can either be provided via the contents of a field or a static value provided below.

File Name Value: Static

File Name

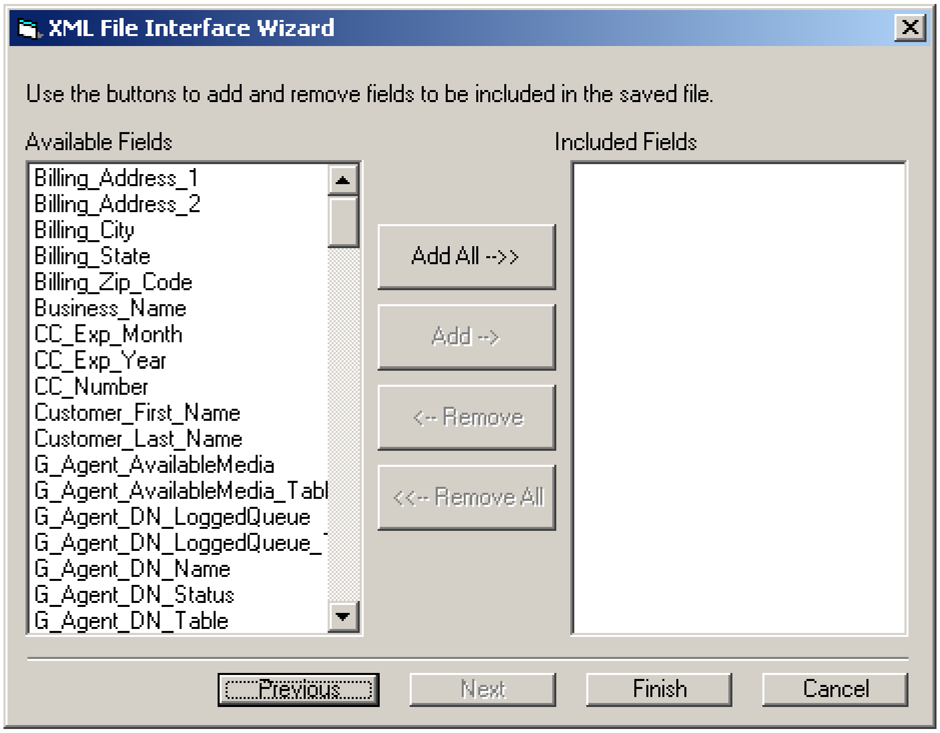
Figure 94: XML File Interface Wizard—First Screen

Table 42describes the controls on the first screen of the XML File Interface Wizard.

Table 42: XML File Interface Wizard—First Screen

Control	Description / What to Do...
Save Interface	Specifies a name for the XML Interface that will save the contents of a field(s) to a file. This name will be shown on the Genesys Agent Scripting button associated with this XML Interface.
Load Interface	Specifies a name for the XML Interface that will load the contents of a flat file into Genesys Agent Scripting. This name will be shown on the Genesys Agent Scripting button associated with this XML Interface.
File Name Value	Use this drop-down box to specify whether the file name will be provided by using the `File Name` box or a Genesys Agent Scripting Field. Valid values are `Static` and `Field`. **Note:** If you select `Static` for the `File Name Value`, multiple users of the runtime generated pages would be writing to the same static file. This can lead to problems.
File Name	Specifies the name of the file.

2. Once you have entered the appropriate information, click `Next` to open the second screen of the `XML File Interface Wizard` (see Figure 95).



Use the buttons to add and remove fields to be included in the saved file.

Billing_Address_1
Billing_Address_2
Billing_City
Billing_State
Billing_Zip_Code
Business_Name
CC_Exp_Month
CC_Exp_Year
CC_Number
Customer_First_Name
Customer_Last_Name
G_Agent_AvailableMedia
G_Agent_AvailableMedia_Tabl
G_Agent_DN_LoggedQueue
G_Agent_DN_LoggedQueue_
G_Agent_DN_Name
G_Agent_DN_Status
G_Agent_DN_Table

Figure 95: XML File Interface Wizard—Second Screen

Table 43describes the controls on the second screen of the `XML File Interface Wizard`.

Table 43: XML File Interface Wizard—Second Screen

Control	Description / What to Do...
Available Fields	Displays the predefined Genesys Agent Scripting Fields.
Add All -->>	Adds all fields in the `Available Fields` window to the `Included Fields` window.
Add -->	Adds the selected field from the `Available Fields` window to the `Included Fields` window.
<-- Remove	Removes the selected field from the `Included Fields` window, and places it back in the `Available Fields` window.

Table 43: XML File Interface Wizard—Second Screen (Continued)

Control	Description / What to Do...
<<-- Remove All	Removes all fields from the `Included Fields` window, and places them back in the `Available Fields` window.
Included Fields	Displays the fields selected to be included in the saved file.

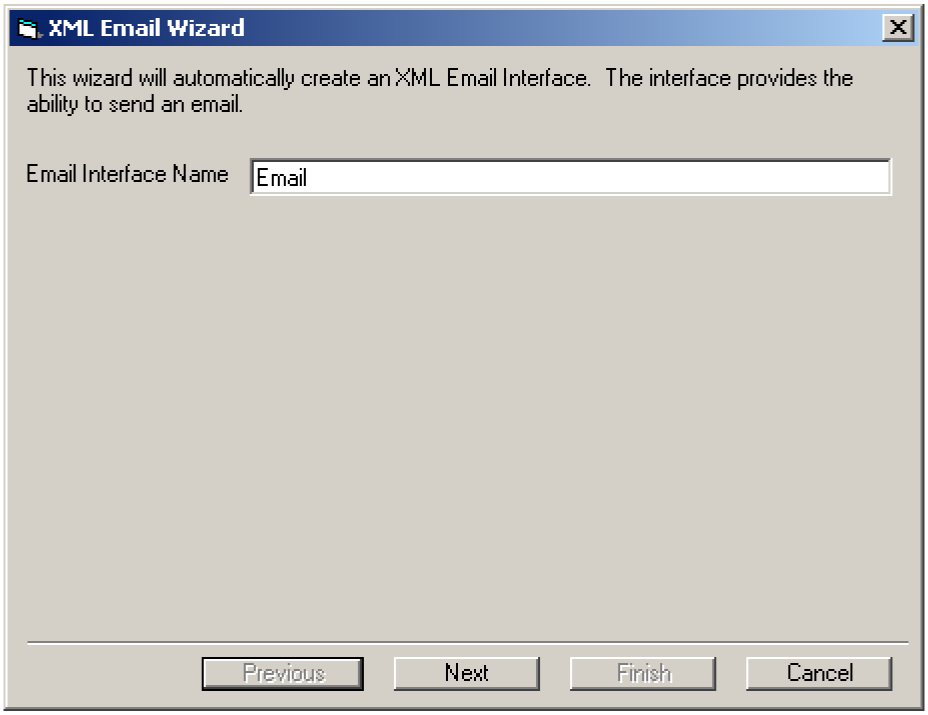
3. Once you have selected the fields that you want to include in the file to be saved, click `Finish`.
   The following message will appear:
   `If you would like to view these interfaces...`
4. To view and edit your new XML Interfaces, leave the `Edit` check boxes selected, click `Close,` and proceed to “Creating an XML Interface—Other” on page 234; otherwise, clear the `Edit` check boxes, and click `Close` to add your interfaces to the `XML Interface List`.

End of procedure

Procedure: Creating an e-mail XML Interface

Start of procedure

1. From the `XML Wizard` (see Figure 93), select `e-mail` to open the first screen of the `XML e-mail Wizard` (see Figure 96).

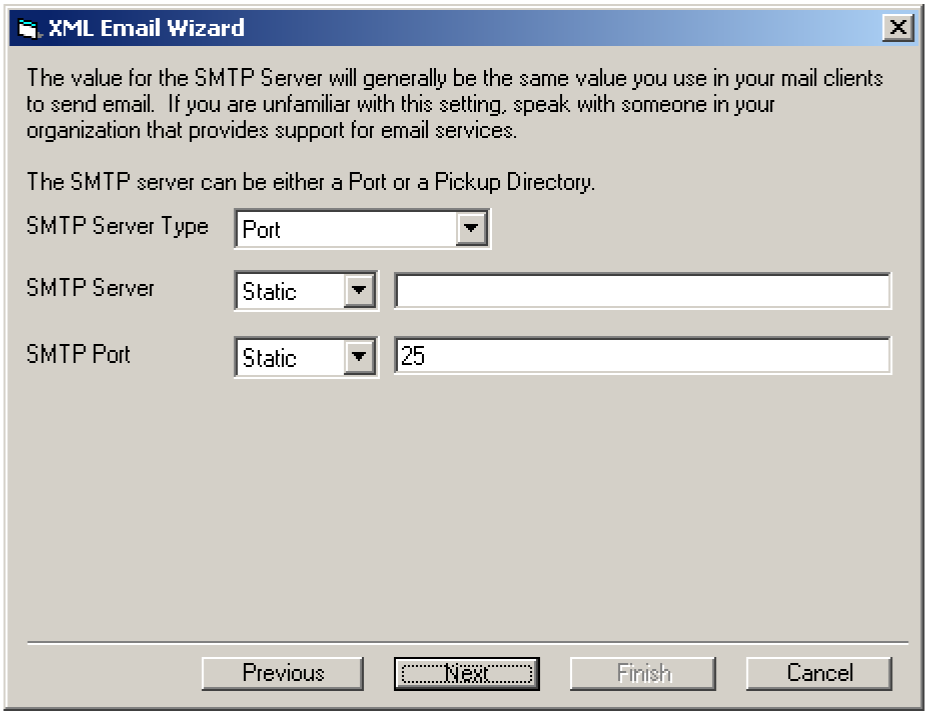


This wizard will automatically create an XML Email Interface. The interface provides the ability to send an email.

Email Interface Name: Email

Figure 96: XML e-mail Wizard—First Screen

2. In the `e-mail Interface Name` box, enter a name for the Interface, and then click `Next` to open the second screen of the `XML e-mail Wizard` (see Figure 97).



The value for the SMTP Server will generally be the same value you use in your mail clients to send email. If you are unfamiliar with this setting, speak with someone in your organization that provides support for email services.

The SMTP server can be either a Port or a Pickup Directory.

SMTP Server Type: Port

SMTP Server: Static

SMTP Port: Static 25

Figure 97: XML e-mail Wizard—Second Screen

Table 44describes the controls on the second screen of the XML e-mail Wizard.

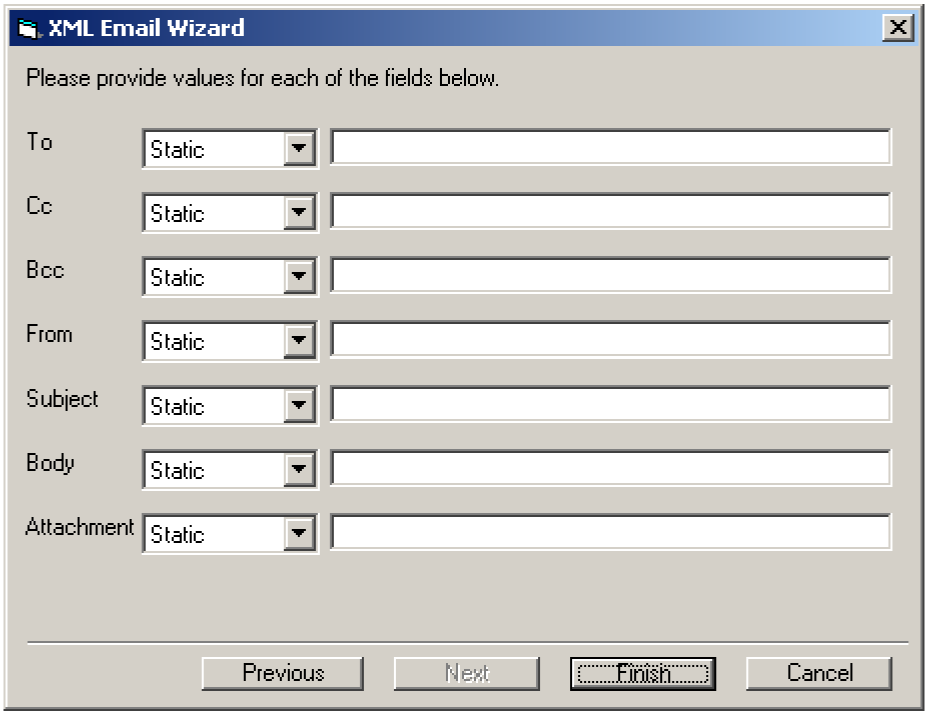
Table 44: XML e-mail Interface Wizard—Second Screen

Control	Description / What to Do...
SMTP Server Type	Specifies the SMTP Server type. Valid values are Port and Pickup Directory.
SMTP Server	When SMTP Server Type is set to Port, this drop-down list is used to specify whether the SMTP Server value will be Static or whether it will come from a predefined Genesys Agent Scripting Field.
SMTP Server Value	Specifies the SMTP Server that you will use to send e-mail.
SMTP Port	When SMTP Server Type is set to Port, this drop-down list is used to specify whether the SMTP Port value will be Static, or whether it will come from a predefined Genesys Agent Scripting Field.
SMTP Port Value	Specifies the SMTP Port to which this Interface will connect.

Table 44: XML e-mail Interface Wizard—Second Screen (Continued)

Control	Description / What to Do...
SMTP Directory	When SMTP Server Type is set to Pickup Directory, this drop-down list is used to specify whether the SMTP Directory value will be Static, or whether it will come from a predefined Genesys Agent Scripting Field.
SMTP Directory Value	Specifies the SMTP directory.

3. Once you have specified the SMTP information, click Next to open the third screen of the XML e-mail Wizard (see Figure 98).



XML Email Wizard

Please provide values for each of the fields below.

To	Static	
Cc	Static	
Bcc	Static	
From	Static	
Subject	Static	
Body	Static	
Attachment	Static	

Figure 98: XML e-mail Wizard—Third Screen

Table 45describes the controls on the third screen of the XML e-mail Wizard. All values can be controlled by using a drop-down list to specify whether the value will be Static or a Genesys Agent Scripting Field.

Table 45: XML e-mail Wizard—Third Screen

Control	Description / What to Do...
To	Specifies the e-mail addresses of one or more recipients who are the primary audience. All recipients can see every address listed in this field.
Cc	Specifies the e-mail addresses of one or more secondary recipients. All recipients can see every address listed in this field.
Bcc	Specifies the e-mail addresses of secondary recipients, but recipients in the Bcc field will see only the see addresses in the To and Cc fields.
From	Specifies the sender’s e-mail address.
Subject	Specifies the subject line or title of the e-mail.
Body	Contains the sender's main message.
Attachment	Contains files to be attached to the email.

4. Once you have specified values for the appropriate fields, click `Finish`. The following message will appear:
   `If you would like to view this interface...`

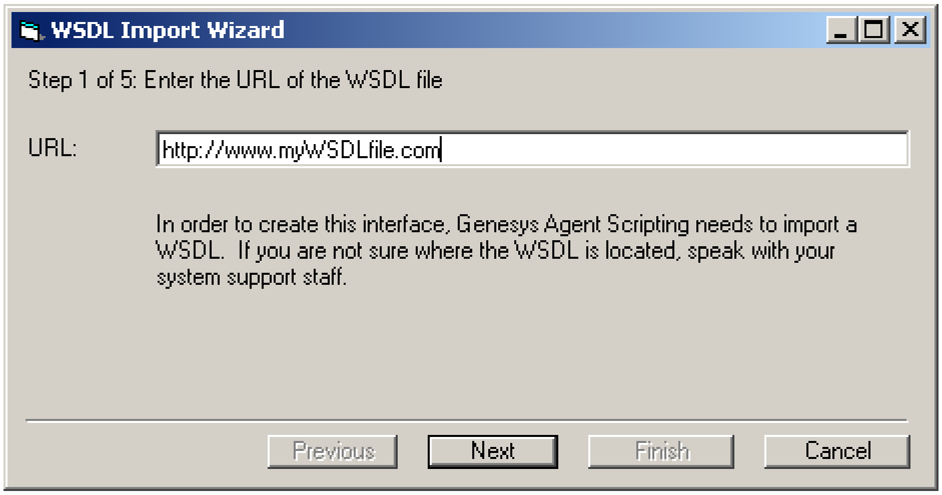
5. To view and edit your new XML Interface, leave the `Edit e-mail Interface` check box selected, click `Close`, and proceed to “Creating an XML Interface—Other” on page 234; otherwise, clear the `Edit e-mail Interface` check box, and click `Close` to add your new interface to the `XML Interface List`.

End of procedure

Procedure: Creating a WSDL XML Interface

Start of procedure

1. From the `XML Wizard` screen (see Figure 93), select `WSDL` to open Step 1 of the `WSDL Import Wizard` (see Figure 99).



Step 1 of 5: Enter the URL of the WSDL file

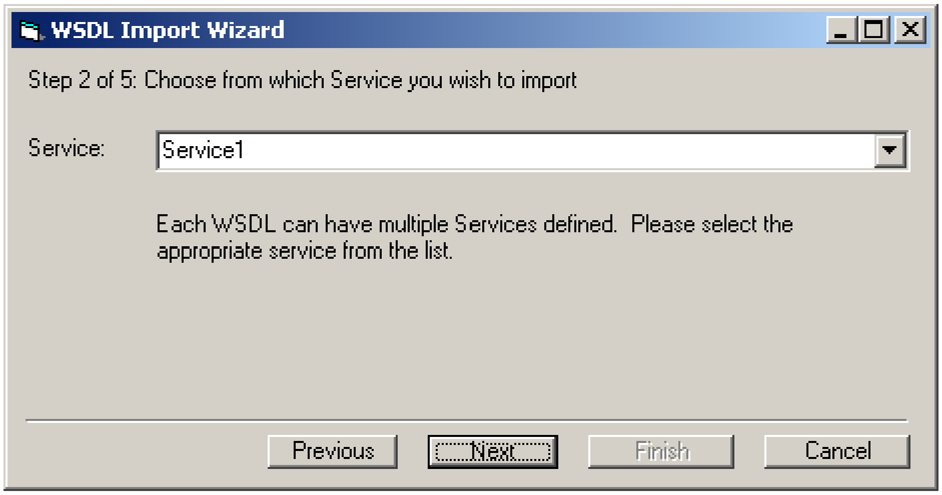
URL: http://www.myWSDLfile.com

In order to create this interface, Genesys Agent Scripting needs to import a WSDL. If you are not sure where the WSDL is located, speak with your system support staff.

Figure 99: WSDL Import Wizard—Step 1

2. In the `URL` box, type the URL of the XML Web Service or the path to the WSDL file, and then click `Next` to open Step 2 of the `WSDL Import Wizard` (see Figure 100).

Note: The URL or path that you enter is the design-time URL. This can be different from the runtime URL.



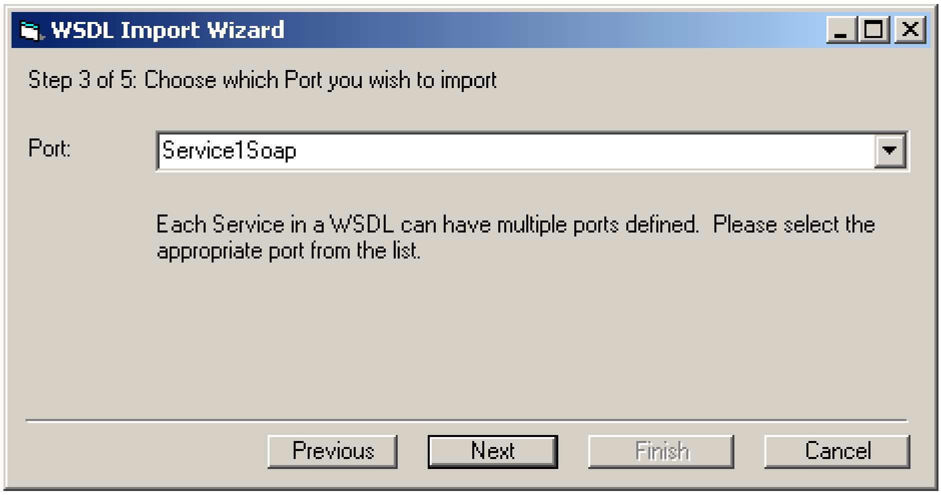
Step 2 of 5: Choose from which Service you wish to import

Service: Service1

Each WSDL can have multiple Services defined. Please select the appropriate service from the list.

Figure 100:WSDL Import Wizard—Step 2

3. From the `Service` drop-down list, select the appropriate Service, and then click `Next` to open Step 3 of the `WSDL Import Wizard` (see Figure 101).



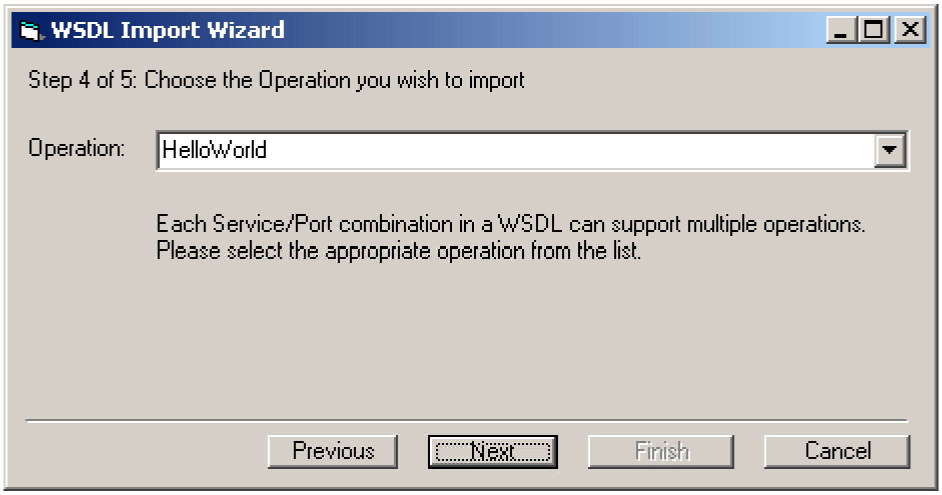
Step 3 of 5: Choose which Port you wish to import

Port: Service1Soap

Each Service in a WSDL can have multiple ports defined. Please select the appropriate port from the list.

Figure 101:WSDL Import Wizard—Step 3

4. From the `Port` drop-down list, select the appropriate port, and then click `Next` to open Step 4 of the `WSDL Import Wizard` (see Figure 102).



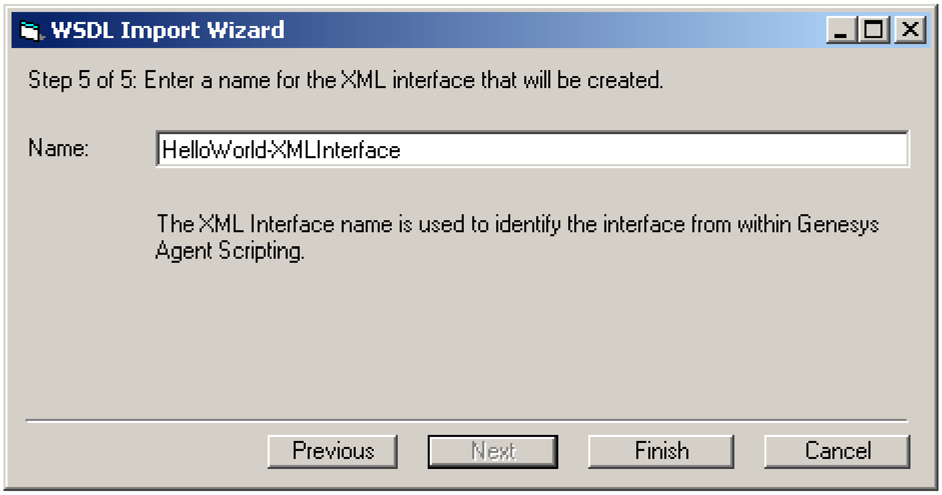
Step 4 of 5: Choose the Operation you wish to import

Operation: HelloWorld

Each Service/Port combination in a WSDL can support multiple operations. Please select the appropriate operation from the list.

Figure 102:WSDL Import Wizard—Step 4

5. From the `Operation` drop-down list, select the appropriate operation, and then click `Next` to open Step 5 of the `WSDL Import Wizard` (see Figure 103).



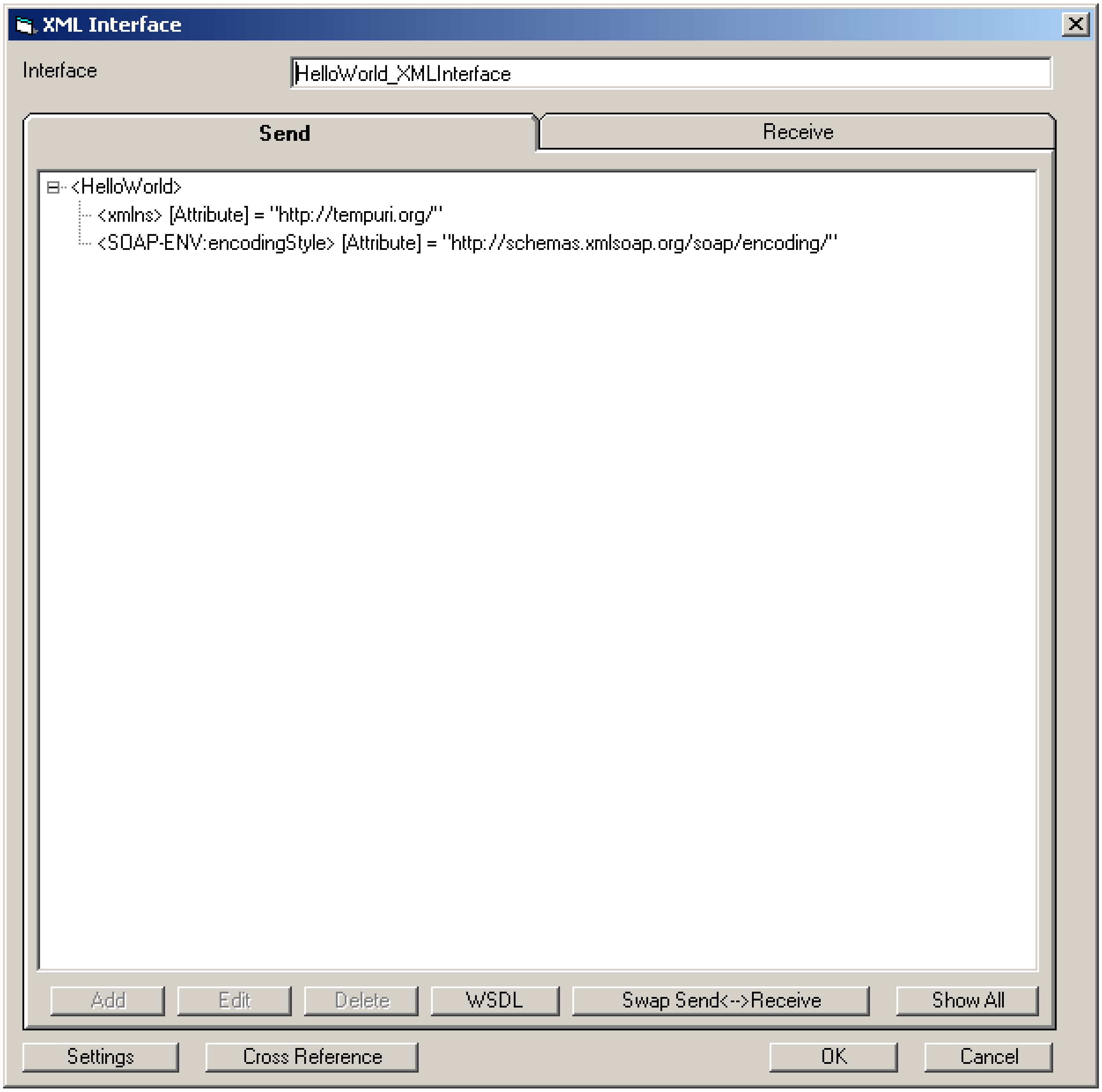
Step 5 of 5: Enter a name for the XML interface that will be created.

Name: HelloWorld-XMLInterface

The XML Interface name is used to identify the interface from within Genesys Agent Scripting.

Figure 103:WSDL Import Wizard—Step 5

6. In the `Name` box, enter a name for the WSDL XML Interface that will be created, and then click `Finish`.
   The following message will appear:
   `If you would like to view this interface...`
7. To view and edit your new WSDL XML Interface, leave the `Edit WSDL Interface` check box selected, and click `Close` to open the `XML Interface` dialog box (see Figure 104); otherwise, clear the `Edit WSDL Interface` check box, and click `Close` to add your interface to the `XML Interface List`.



Interface HelloWorld_XMLInterface

```
⊟ <HelloWorld>
    <xmlns> [Attribute] = "http://tempuri.org/"
    <SOAP-ENV:encodingStyle> [Attribute] = "http://schemas.xmlsoap.org/soap/encoding/"
```

Figure 104:XML Interface Dialog Box—WSDL

End of procedure

Procedure:
Creating an XML Interface—Other

Start of procedure

1. From the XML Wizard screen (see Figure 93), select Other to open the XML Interface dialog box (see Figure 105).

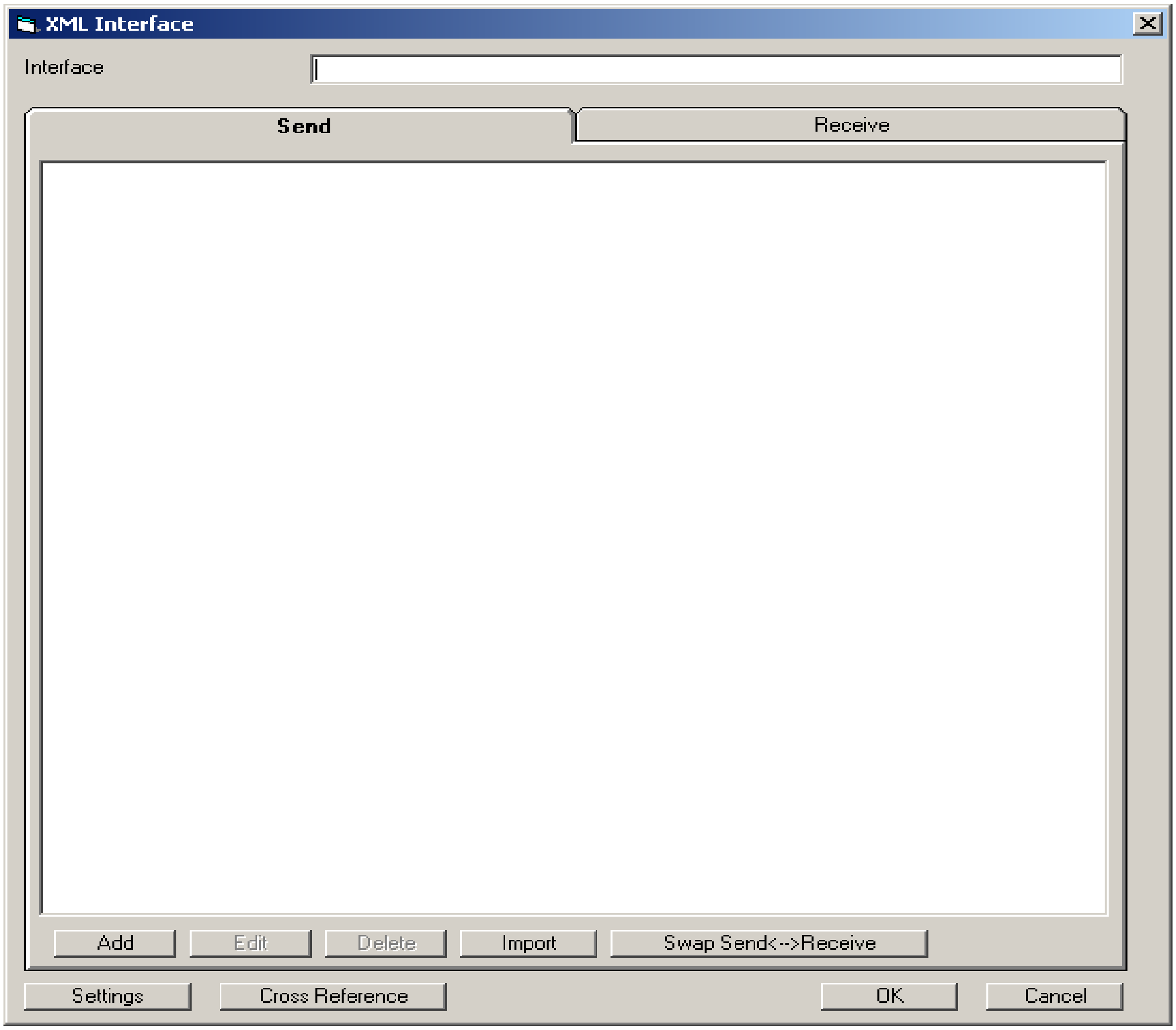


Figure 105:XML Interface Dialog Box

Selecting Other enables you to create an XML Interface to communicate with other systems through XML. Table 46describes the controls on the XML Interface dialog box.

Table 46: XML Interface Dialog Box

Control	Description / What to Do...
Interface	Specifies the name of the XML Interface.
Send Tab	Displays the XML tags that are defined for the post of the XML Interface data.
Receive Tab	Displays the XML tags that are defined for the post of the XML Interface data.
Add	Use this button to add a new tag. The new tag will be added as a child to the tag currently selected. Clicking the Add button opens the XML Tag dialog box (see Figure 106).
Edit	Use this button to modify a selected tag on the Send and Receive tabs. Clicking the Edit button opens the XML Tag dialog box (see Figure 106).
Delete	Deletes the selected XML tag. **Warning!** When you delete a tag, all of its child tags are also deleted.
WSDL	Use this button to open the WSDL Import Wizard (see Figure 99). This button is present only if the interface that you are editing is of type WSDL.
Import	Enables you to import a .dtd or .xsd file. When you click Import, a Windows Open File dialog box appears. Navigate to the .dtd or .xsd file for this XML Interface, and open it. The tags that are defined in the .dtd or .xsd file, and many of their options, will automatically be set up in Genesys Agent Scripting. A .dtd file defines the XML tags for the interface. An .xsd file specifies how to formally describe the elements in an XML document. This description can be used to verify that each item of content in a document adheres to the description of the element in which the content is to be placed.
Swap Send<-->Receive	Causes the contents of the Send tab to switch places with the contents of the Receive tab.

Table 46: XML Interface Dialog Box (Continued)

Control	Description / What to Do...
Settings	Opens the `XML Settings for NewXMLTags` dialog box. See “Defining XML Settings for NewXMLTags” on page 239 for details.
Cross Reference	Enables you to see where an XML Interface is being used. Clicking the `Cross Reference` button opens the `Object Cross Reference` dialog box.

End of procedure

Procedure: Creating XML Tags

Start of procedure

1. Use the XML Tag dialog box to create XML tags (see Figure 106).

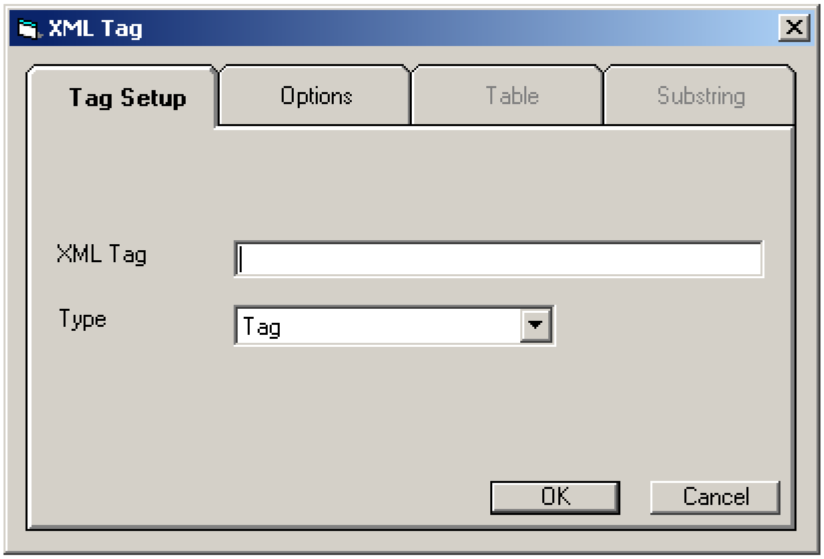


Figure 106:XML Tag Dialog Box

Table 47describes the controls on the `XML Tag` dialog box.

Table 47: XML Tag Dialog Box

Control	Description / What to Do...
Tag Setup Tab	This tab is used to set up the tag.
XML Tag	Specifies the name of the tag for the XML request or response. The tag name is case sensitive.
Type	Specifies the type of XML tag. Valid values in the drop-down list are: • Tag—Used to supply an XML tag with no value. • Filter—In order for this option to appear, you must define an attribute (by checking the Attribute check box on the Options tab) while on the Receive tab of the XML Interface dialog box, and assign it a static value. Genesys Agent Scripting will filter the return values, looking for that attribute. Then, you can set up the response to load different fields, based on the attribute's value. • Static—Used to supply an XML tag with a static value. It appears only for the Send tab. • Field—Used to supply an XML tag with a Genesys Agent Scripting Field value that is not a table. The Field's value will be sent as the value of a tag for a send, and it will be set in the event of a receive. • Table—Used when the tag's value is from a Genesys Agent Scripting Table. • Column—This option appears when the tag is a child of a tag of type Table. You can then assign the value to one of the columns in the parent table.
Value	Specifies the Field or Table that the tag will be assigned. This drop-down list appears only if Type is set to Field or Table.
Options Tab	Displays the options that are available for the tag.
Attribute	Select this check box if this tag is an attribute for its parent tag, rather than a child tag.
Optional	Select this check box if this tag is optional. An optional tag will be sent only if it has a value.
Optional if any child missing	Enables entire parent tag to be ignored if one of its children is missing its value.

Table 47: XML Tag Dialog Box (Continued)

Control	Description / What to Do...
Enclose value with CDATA indicator	If this option is selected, this tag's value will be enclosed with `CDATA` indicators. This means that everything between the `CDATA` indicators will be treated as raw data rather than XML.
Table tab	Used to set row and column values for the tag. This tab is enabled only when the tag `Type` on the `Tag Setup` tab is set to `Table,` and a table has been selected as the tag `Value`.
Row	Applies only to repeating groups for tags on the Send tab. Valid values are: • All Rows—If all rows are selected, then all rows in a table will be sent. • Current Row—Only the current row, or the last row that the user selected, will be sent. • Selected Rows—All selected rows in a multi-row table will be sent.
Select First Row on returned values	If selected, the first row in the table will be selected by default when the table is loaded.
Assign value to column	This option is important if the field for this tag points to a table. It indicates that there are no child tags to this tag. This option is used in combination with the Column option. Set the `Column` option to point at a column in this field's table to determine the value for this tag.
Column	This option is important if the field for this tag points to a table. It is used in combination with the `Assign value to column` option. See the description of the `Assign value to column` option for information about how to use this option.
Substring Tab	Used to create substrings for fields.
Concatenate Children	Select this check box if you want to join child tags together.
Substring Frame	Substrings can be used to output partial values of a parent tag. For example, suppose that the parent tag of this tag points to a telephone number. The substring start and length properties can be used to output parts of the telephone number field.

Table 47: XML Tag Dialog Box (Continued)

Control	Description / What to Do...
Start	Points to the position in the string at which you want the substring to start.
Length	Tells Genesys Agent Scripting the number of characters to place in this value. If you leaves the `Length` value blank, Genesys Agent Scripting will output the value, starting from the `Start` value and continuing to the end of the string.

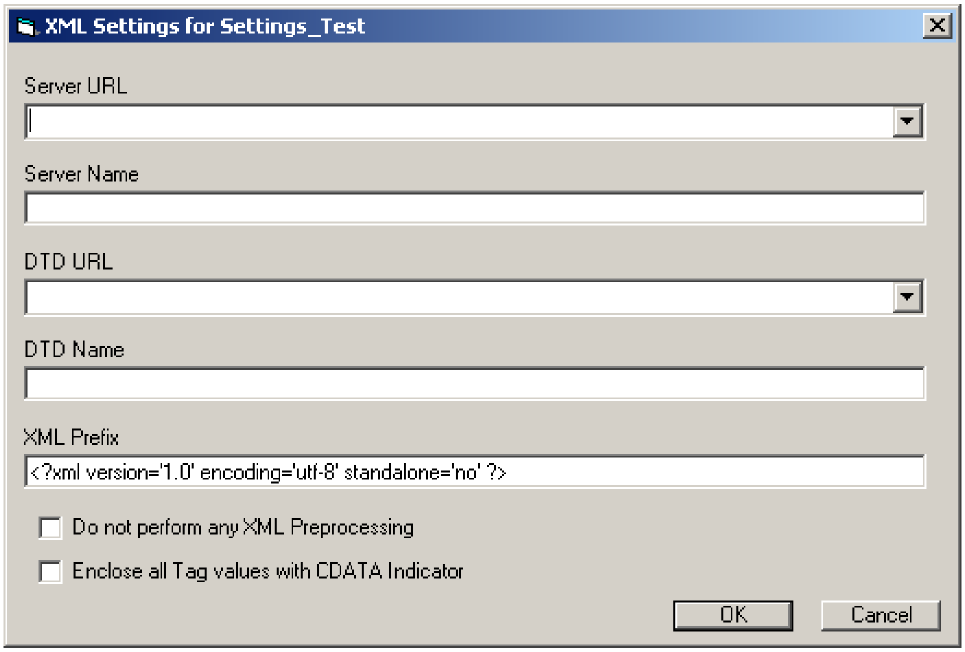
2. Click OK to save your choices or Cancel to abandon them.

End of procedure

Procedure: Defining XML Settings for NewXMLTags

Start of procedure

1. In the `XML Interface` dialog box (see Figure 105), enter a name for your new Interface, and then click the `Settings` button to open the `XML Settings` dialog box (see Figure 107).



<?xml version='1.0' encoding='utf-8' standalone='no' ?>

☐ Do not perform any XML Preprocessing

☐ Enclose all Tag values with CDATA Indicator

Figure 107:XML Settings Dialog Box

Table 48describes the controls on the `XML Settings` dialog box.

Table 48: XML Settings Dialog Box

Control	Description / What to Do...
Server URL	Specifies the address of the XML server that will handle this XML post. It is the beginning part of the URL. The drop-down list is populated with Genesys Agent Scripting configuration file fields. Select the field that contains the Server URL. It is not mandatory; however, if it is left blank, `Server Name` must contain the fully qualified URL for the XML server. **Note**: The URL that you specify is the runtime URL. This can be different from the design-time URL (which is specified using the WSDL Import Wizard).
Server Name	Specifies the name of the XML server that will handle this XML post. It should be either the file name of the server or the fully qualified URL—for example: `http://<servername> or <IP address>/website/WWGEMail.asp` This field is mandatory.
DTD URL	Specifies the location of the definition file to be used for the XML post, if one is required. The drop-down list is populated with Genesys Agent Scripting configuration file fields. Select the field that contains the DTD URL. It is not mandatory; however, if it is left blank, `DTD Name` must contain the fully qualified URL for the DTD.
DTD Name	Specifies the name of the definition file to be used for the XML post, if a DTD is required.
XML Prefix	Specifies the XML declaration that appears at the top of an XML file. If the file has a prefix, it must contain ?xml, the XML version, the type of encoding, and whether the document needs an outside DTD.

Table 48: XML Settings Dialog Box (Continued)

Control	Description / What to Do...
Do not perform any XML Preprocessing	If this check box is selected, Genesys Agent Scripting will not perform any XML preprocessing for this interface. This refers to the global preprocessing that occurs from the `WWGXMLPost.inc` file.
Enclose all Tag values with CDATA indicator	If this check box is selected, all tag values will be enclosed with `CDATA` Indicators. This means that everything between the `CDATA` indicators will be treated as raw data rather than XML.

2. Click OK to save your choices or Cancel to abandon them.

End of procedure

Example

This section provides step-by-step examples that illustrate how to set up XML Interfaces.

Setting Up a File Save and Load XML Interface

Let's create a File Save and Load XML Interface that will save fields and their contents to a flat file, and load these same fields and their contents from the flat file into Genesys Agent Scripting.

The Fields Used in This Example

The following fields will be used in this example:

- Customer_Billing_List
- Customer_First_Name
- Customer_Last_Name
- Billing_Address_1
- Billing_City
- Billing_State
- Billing_ZIP_Code

The Customer_Billing_List field contains the file name used during execution of the Save and Load interfaces. The other fields are self explanatory. These fields can be created under `Define > Fields`. For details on creating fields, refer to "Creating Fields" on page 65.

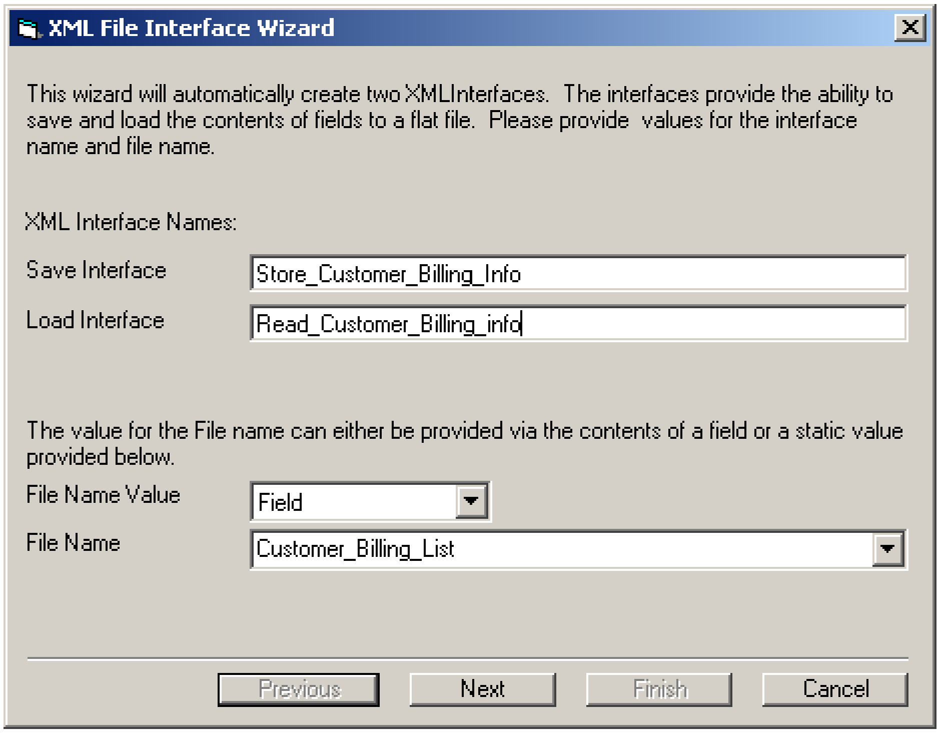
Procedure: Creating the Interfaces

Start of procedure

1. Select `Define > XML Interfaces`.
2. In the `XML Interface List` dialog box, click `Add`.
3. In the XML Wizard, select `File Save and Load`, and then click `OK`.
4. In the `Save Interface` and `Load Interface` boxes, specify names for the corresponding Interfaces.
5. From the `File Name Value` drop-down list, select `Field`.
6. From the `File Name` drop-down list, select the field that you created containing the name of the file that you will be using.

   In this example, our field name is stored in the predefined field called `Customer_Billing_List`.

   Your `XML File Interface Wizard` screen should look similar to Figure 108.



This wizard will automatically create two XMLInterfaces. The interfaces provide the ability to save and load the contents of fields to a flat file. Please provide values for the interface name and file name.

XML Interface Names:

Save Interface: Store_Customer_Billing_Info

Load Interface: Read_Customer_Billing_info

The value for the File name can either be provided via the contents of a field or a static value provided below.

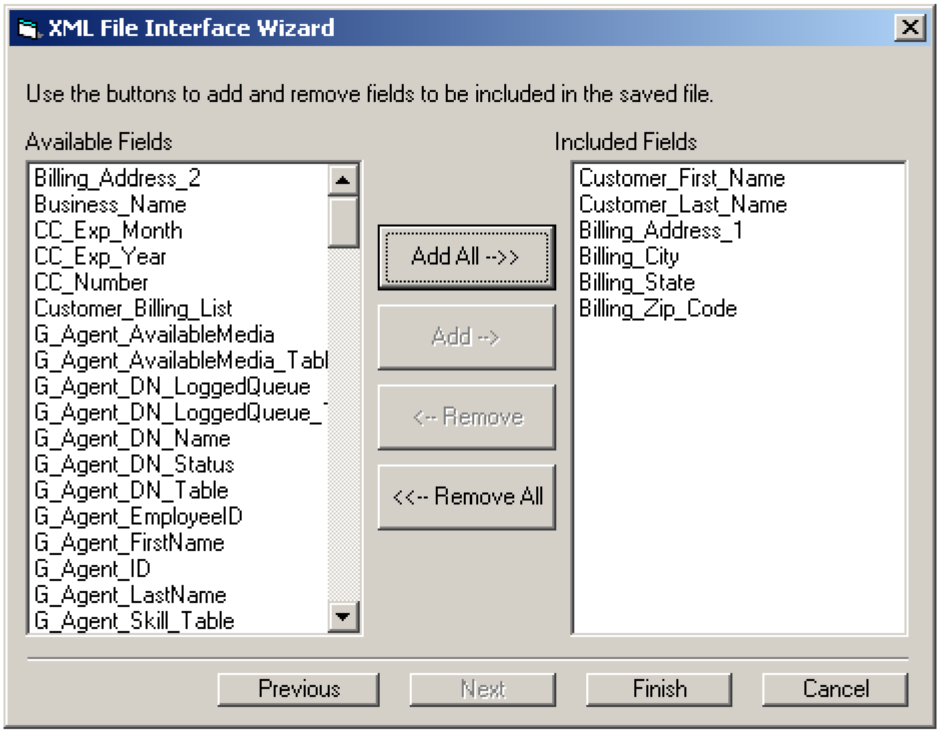
File Name Value: Field

File Name: Customer_Billing_List

Figure 108:XML File Interface Wizard—Example

7. Click **`Next`** to open the second screen of the **`XML File Interface Wizard`**.
8. In the **`Available Fields`** windows, select the fields that you created, and then click **`Add`** to add them to the **`Included Fields`** window.

In this example, we are selecting Customer_First_Name, Customer_Last_Name, Billing_Address_1, Billing_City, Billing_State, and Billing_ZIP_Code as shown in Figure 109.



XML File Interface Wizard

Use the buttons to add and remove fields to be included in the saved file.

Available Fields

Billing_Address_2
Business_Name
CC_Exp_Month
CC_Exp_Year
CC_Number
Customer_Billing_List
G_Agent_AvailableMedia
G_Agent_AvailableMedia_Tabl
G_Agent_DN_LoggedQueue
G_Agent_DN_LoggedQueue_
G_Agent_DN_Name
G_Agent_DN_Status
G_Agent_DN_Table
G_Agent_EmployeeID
G_Agent_FirstName
G_Agent_ID
G_Agent_LastName
G_Agent_Skill_Table

Included Fields

Customer_First_Name
Customer_Last_Name
Billing_Address_1
Billing_City
Billing_State
Billing_Zip_Code

Figure 109:XML File Interface Wizard Second Screen—Example

9. Click Finish, and then click the Close button.
10. In the XML Interface dialog box, verify the resulting XML and then click OK to add your new Interfaces to the XML Interface List.

    In this example, the Interfaces named Store_Customer_Billing_Info and Read_Customer_Billing_Info are added to the XML Interface List.
11. In the Page Editor, open the page to which you want to add your XML File Save and Load Interfaces.
12. From the Action List, select your newly created XML File Save and Load Interfaces, and then click Add to Page.

End of procedure

Chapter

10 Async Interfaces

This chapter provides an overview of how to create and use an Async Interface.

The information in this chapter is divided among the following topics:

What Is an Async Interface?

An Async Interface enables users to develop pages that can receive dynamic data updates from external applications, without refreshing the page.

The Async Interface outlines how XML tags are mapped to Agent Scripting fields. The external application must generate the XML stream with the agreed-to tags, and send it to the Agent Scripting Async Event Dispatcher. This, in turn, will communicate with the running Agent Scripting pages to update the defined fields.

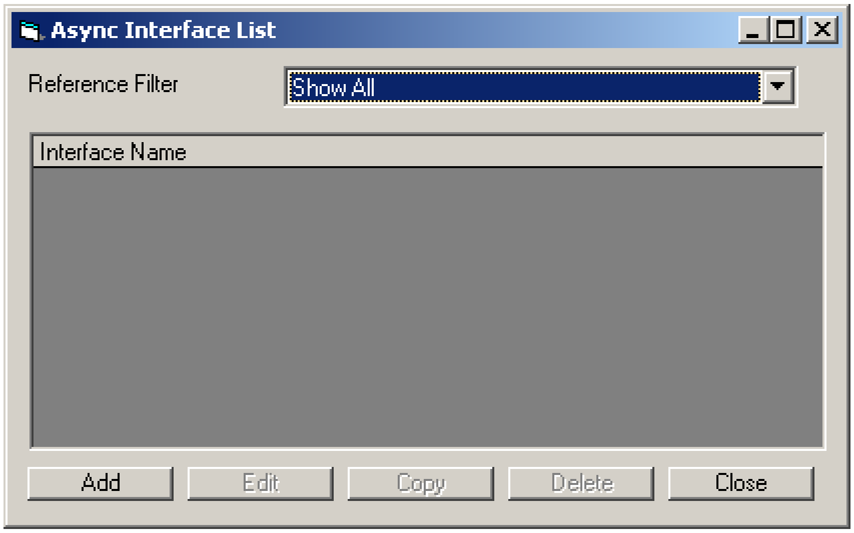
Creating an Async Interface

You will use the `Define` menu.

Procedure:
Create a New Async Interface

Start of procedure

1. Select Define > Async Interfaces to open the Async Interface List dialog box (see Figure 110). From the Async Interface List dialog box, you can edit, copy, delete, or add Async Interfaces.



Reference Filter: Show All

Figure 110:Async Interface List Dialog Box

2. Click Add to initiate the process for creating an Async Interface.

   The Async Interface dialog box opens (see Figure 111).

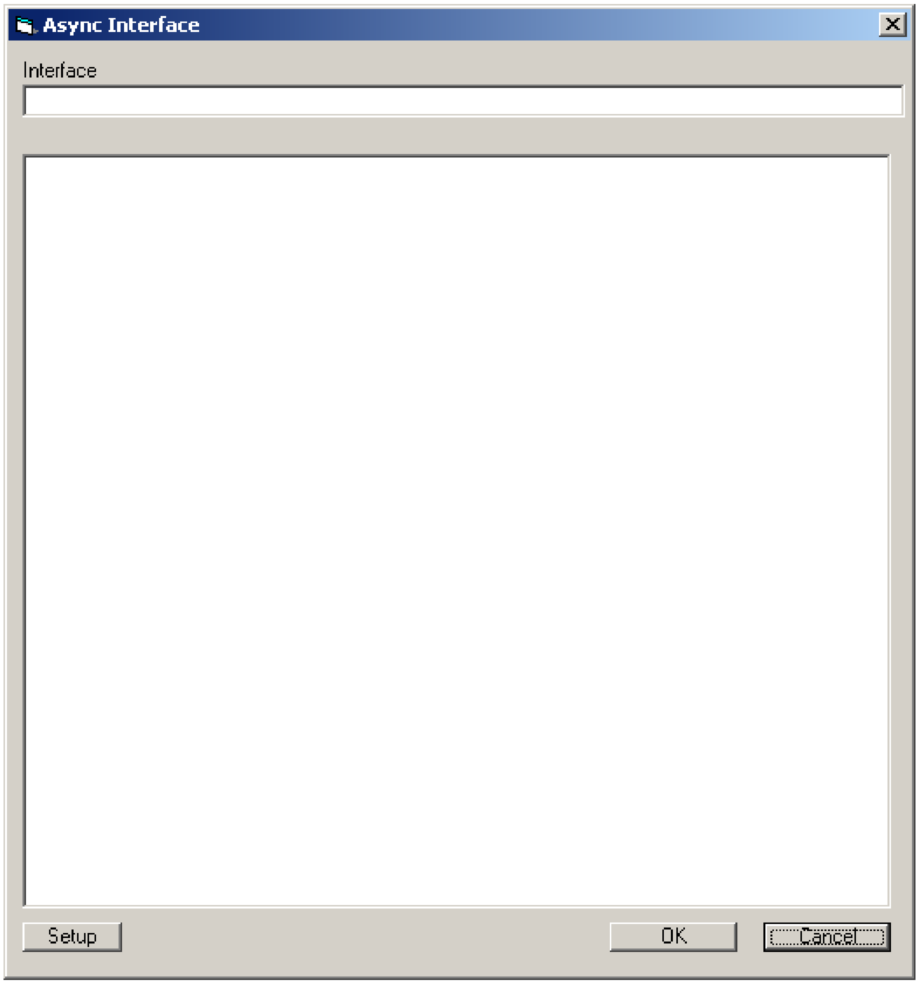
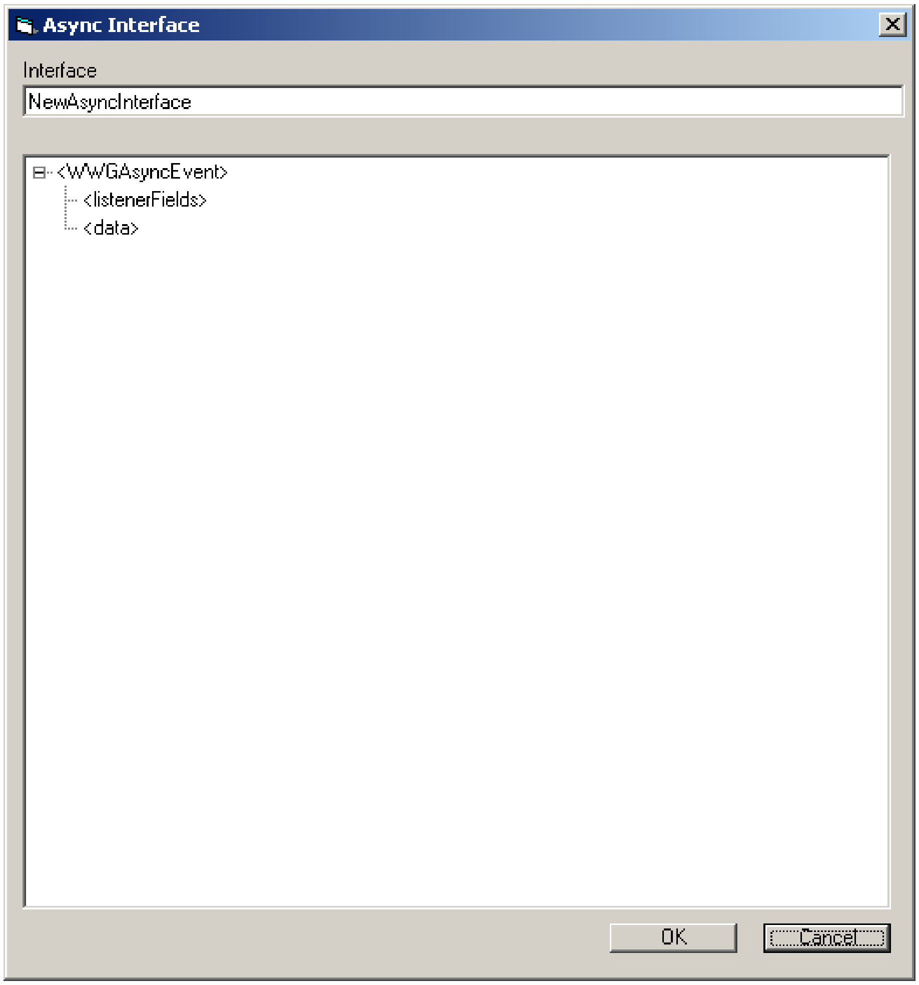


Figure 111:Async Interface Dialog Box

3. In the `Interface` box, type a name for the Interface and then click `Setup`. If you are prompted for a valid name to be generated, click `Yes,` and then click `Setup` again.

   An outline of an Async Interface will be created (see Figure 112).



NewAsyncInterface

```
<WWGAsyncEvent>
    <listenerFields>
    <data>
```

Figure 112:Async Interface Dialog Box—Setup

4. You can now assign the appropriate listener and data fields to the Interface.

- To add an Async listener field, proceed to the "Adding an Async Listener Field" section.
- To add data, proceed to the "Adding Data" section.

End of procedure

Procedure: Adding an Async Listener Field

The `<listenerFields>` tag identifies one or more listener fields that will receive dynamic updates from an external application. Dynamic data assigned to a listener field is stored as a string. To convert this string data to another Agent Scripting field of a different type, you must invoke the Async Interface

PULL VALUE function. This function can be invoked during a page load, or as part of the code for an action.

Start of procedure

1. In the Async Interface dialog box, select the <ListenerFields> tag. An Add button will appear at the bottom of the dialog box.
2. Click Add.

   The Async Listener Field dialog box opens (see Figure 113).

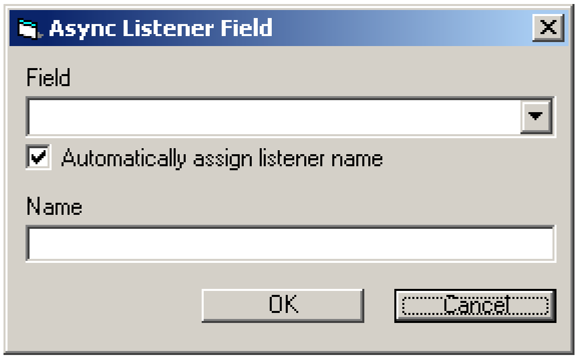


Figure 113:Async Listener Field Dialog Box

Table 49describes the options in the Async Listener Field dialog box.

Table 49: Async Listener Field Properties

Control	Description / What to Do...
Field	Select the appropriate field from the drop-down list. The drop-down list contains fields of the following types: listener, string, multiline text, numeric, and date and time.
Automatically assign listener name	Select this check box if you want the Field name to be used in the Name box.
Name	This is the tag name as it appears in the Async Interface.

3. Once you have completed the entries in the Async Listener Field dialog box, click OK. You are returned to the Async Interface dialog box.
4. To add data to your Async Interface, proceed to the “Adding Data” section, otherwise, click OK to add your Async Interface to the Async Interface List.

End of procedure

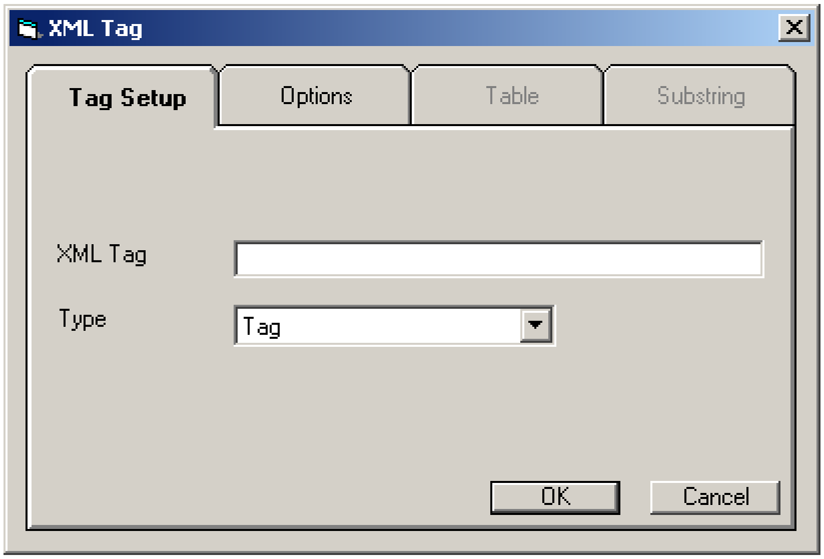
Procedure:
Adding Data

Start of procedure

The <data> tag identifies one or more Agent Scripting fields of any type that will receive dynamic data updates from an external application.

1. From the Async Interface dialog box, select the <data tag>. An Add button will appear at the bottom of the dialog box.
2. Click Add.

The XML Tag dialog box opens with the Tag Setup tab displayed (see Figure 114).



XML Tag

Type: Tag

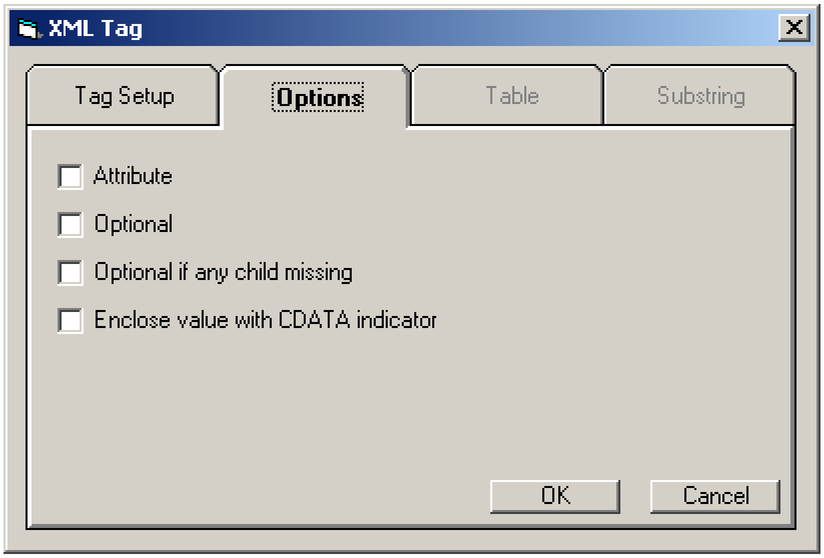
Figure 114:XML Tag Dialog Box—Tag Setup Tab

Table 50describes the parameters on the Tag Setup tab of the XML Tag dialog box.

Table 50: XML Tag Dialog Box—Tag Setup Tab

Control	Description / What to Do...
XML Tag	Specifies the name of the tag for the XML request or response. The tag name is case sensitive.
Type	Specifies the type of XML tag. Valid values in the drop-down list are: • Tag—Used to supply an XML tag with no value. • Filter—In order for this option to appear, you must define an attribute (by checking the Attribute check box on the Options tab) while on the Receive tab of the XML Interface dialog box, and assign it a static value. Genesys Agent Scripting will filter the return values, looking for that attribute. Then, you can set up the response to load different fields, based on the attribute's value. • Static—Used to supply an XML tag with a static value. It appears only for the Send tab. • Field—Used to supply an XML tag with a Genesys Agent Scripting Field value that is not a table. The Field's value will be sent as the value of a tag for a send, and it will be set in the event of a receive. • Table—Used when the tag's value is from a Genesys Agent Scripting Table. • Column—This option appears when the tag is a child of a tag of type Table. You can then assign the value to one of the columns in the parent table.
Value	The Value field will be either a text box or drop-down list, depending on the type of tag selected from the Type drop-down list. The Value field will not appear at all if Type is set to Tag. • If Type is set to Filter, type a value in the Value text box. • If Type is set to Field, use the Value drop-down list to select a Field in which to store the data. • If Type is set to Table, use the Value drop-down list to select from among the existing Table fields.
OK	Click OK when the mapping has been set. Agent Scripting returns you to the Async Interface dialog box.

3. To define options for the tag, click the `Options` tab (see Figure 115).



□ Attribute

□ Optional

□ Optional if any child missing

□ Enclose value with CDATA indicator

Figure 115:XML Tag Dialog Box—Options Tab

Table 51describes the parameters on the `Options` tab of the `XML Tag` dialog box.

Table 51: XML Tag Dialog Box—Options

Control	Description / What to Do...
Attribute	Select this checkbox if this tag is an attribute for its parent tag, rather than a child tag.
Optional	An optional tag will be sent only if it has a value.
Optional if any child missing	Select this check box to enable an entire parent tag to be ignored if one of its children is missing its value.
Enclose value with CDATA indicator	If this option is selected, this tag's value will be enclosed with `CDATA` indicators. This means that everything between the `CDATA` indicators will be treated as raw data rather than XML.

For information on the `Table` and `Substring` tabs of the `XML Tag` dialog box, refer to "Creating XML Tags" on page 236.

4. Once you have completed the entries in the `XML Tag` dialog box, click `OK`. You are returned to the `Async Interface` dialog box.

   To add listener fields to your Async Interface, go to "Adding an Async Listener Field" on page 248; otherwise, save your changes and exit.

5. Click OK to add your Async Interface to the Async Interface List.

End of procedure

External Application Responsibilities

Using the specification from the Async Interface, the external application constructs an xml data stream, along with the following tags:

- <asyncInterfaceName>—Identifies the name of the Async Interface. The external application uses this tag to identify which Async Interface it is using.
- <asyncClientID>—Identifies the numeric ID for the client running the generated Web application. This ID is automatically created when the generated Web application is started. The external application specifies this tag to send data to a specific client. The Agent Scripting-generated Web application can retrieve the value of the Async Client ID through the Async Interface GET ASYNC CLIENT ID function, which assigns the value to an Agent Scripting Field. This type of function can be invoked during a page load or as part of the code for an action. The Client ID must be sent to the external application in order for the application to identify how to send data to a specific client.
- <subject>—Identifies a subject to which a client can subscribe in order to receive dynamic data updates from an external application. The external application sets this tag to a specific subject name for the data being updated. The Agent Scripting-generated Web application can subscribe to the subject through the Async Interface SUBSCRIBE TO MESSAGE function, which subscribes to a specific subject. This type of function can be invoked during a page load, or as part of the code for an action.

Note: If the external application specifies an <asyncClientID> tag along with this <subject> tag, then the <subject> tag is ignored.

The external application issues an HTTP post to the Agent Scripting Async Event Dispatcher, sending it the XML data stream. The file name of the Agent Scripting Async Event Dispatcher is WWGAsyncEventDispatcher (.jsp or .aspx), and it is located where the generated Agent Scripting pages are deployed.

After the external application issues the HTTP post to the Agent Scripting Async Event Dispatcher, an XML data stream containing the results is returned as the response to the HTTP post request. The following outlines the XML data stream in the response:

```
<WWGAsyncEventResponse/>
   <description/>
   <resultCode/>
   <asyncClientResult/>
```

```
<asyncClientID/>
<resultCode/>
<description/>
```

If the request was to send the data directly to a specific client, there will be only one `<asyncClientResult>` tag returned for the client. If the request was to broadcast the data to any clients who have subscribed on the specified subject, there will be an `<asyncClientResult>` tag for each client.

Example

The following is an example of an Async Interface. Its name is `myAsyncInterface`, and it maps data to two listener fields, three string fields, and one or more rows of a table:

```
<WWGAsyncEvent>
  <asyncClientID>
  <asyncInterfaceName>
  <listenerFields>
    listenerField1 = <listener name=myListenerName1>
    listenerField2 = <listener name=myListenerName2>
  <data>
    stringField1 = <field1Tag>
    stringField2 = <field2Tag>
    stringField3 = <field3Tag>
    tableField = <tableTag> {Repeating]
      col1Field = <col1Tag>
      col2Field = <col2Tag>
      col3Field = <col3Tag>
```

Mapping data to a `Table` field requires that you identify the columns of interest. This is done by selecting the table entry and adding additional sub-elements, one for each column, to identify the name for the column's XML tag and the table column (an Agent Scripting Field) to which it is to be mapped. The external application can then send multiple records to the table. In the example, three table columns have been specified.

The external application uses the preceding Async Interface specification to construct an XML data stream that will be sent to the Agent Scripting Async Event Dispatcher.

The following is an example of an XML data stream. The external application is sending the data to a specific client. The client must previously have sent its `Async Client ID` to the external application. The data being sent consists of five Agent Scripting Fields (three `String` fields and two `Listener` fields), and three `Table` records.

```
<WWGAsyncEvent>
  <asyncClientID>22</asyncClientID>
  <asyncInterfaceName>myAsyncInterface</asyncInterfaceName>
  <listenerFields>
    <listener name='myListenerName1'>listenerValue1</listener>
    <listener name='myListenerName2'>listenerValue2</listener>
  </listenerFields>
  <data>
    <field1Tag>stringValue1</field1Tag>
    <field2Tag>stringValue2</field2Tag>
    <field3Tag>stringValue3</field3Tag>
    <tableTag>
      <col1Tag>Rec1Column1Value</col1Tag>
      <col2Tag>Rec1Column2Value</col2Tag>
      <col3Tag>Rec1Column3Value</col3Tag>
      <col1Tag>Rec2Column1Value</col1Tag>
      <col2Tag>Rec2Column2Value</col2Tag>
      <col3Tag>Rec2Column3Value</col3Tag>
      <col1Tag>Rec3Column1Value</col1Tag>
      <col2Tag>Rec3Column2Value</col2Tag>
      <col3Tag>Rec3Column3Value</col3Tag>
    </tableTag>
  </data>
</WWGAsyncEvent>
```

If the external application wants to broadcast the data to any clients who previously subscribed on a specific subject name, the external application removes the `<asyncClientID>` tag and replaces it with the `<subject>` tag specifying the name of the subject to broadcast.

We have described how a generated Agent Scripting Web application can receive dynamic data updates from an external application. In this case, the external application sends data to the client Web application whenever the data is modified without the client having to issue a request for the data.

However, what if we want the client's Web application to request data that might take some time to process? We would prefer not to have to block the client's Web application until the data arrives. If you construct a request (through an Action push button) to an XML Interface, the user cannot issue any additional requests until the current request is completed. Depending on the XML Interface this might take some time. However, with the Async Interface, the request to an XML Interface can run in the background, enabling you to continue issuing other requests. The response from the XML Interface is returned through the Async Event Dispatcher Interface.

In order to have an XML Interface execute a request in the background, create an XML Interface to the XML Async Interface. This interface takes as input the actual XML Interface that is to run in the background, along with the Async Event Dispatcher Interface, in order to map any response data to Agent Scripting Fields. The XML Async Interface is the file called `WWGAsyncXML` (.jsp or .aspx), which resides wherever the client's Web application is deployed. You can create an XML Interface to this file by providing the following XML data stream:

```
<WWGAsyncXML/>
   <asyncClientID/>
   <asyncInterfaceName/>
<data/>
   <listenerFields/>
   <url/>
   <aedUrl/>
```

The tags <asyncClientID>, <asyncInterfaceName>, <data>, and <listenerFields> are the same ones an external application uses to send data to a client's Web application through the Async Event Dispatcher Interface. In this case, the external application is the XML Async Interface. The XML Async Interface cannot take as input the <subject> tag. Only one <asyncClientID> tag must be specified. In addition to these tags, the following two tags are required:

- <url>—Identifies the URL for the XML Interface that is to execute requests in the background.
- <aedUrl>—Identifies the URL for the Async Event Dispatcher Interface that is to send the response back to the client's Web application.

Genesys

Chapter

11 Adding Branching Logic

This chapter provides an overview of the process of adding branching logic within a script.

The information in this chapter is divided among the following topics:

What Are Branches?

Branches are a set of logic that directs the flow from one Page to the next within the various Process Flows. Branches provide the map through an interaction that is followed when Action buttons are clicked.

Branching is the process of setting up the navigational flow between Pages.

Branching was also described earlier in the context of the Branch Action type. See “Descriptions of Action Types” on page 112, “Navigation Actions” on page 114, “Branch” on page 113, and “Branch Commands” on page 137 for examples of how branching can be used.

Types of Branches

There are currently four kinds of branches that you can create within the Genesys Agent Scripting Development Environment:

- Always Branch
- Conditional
- Return
- Stop

Always Branch

If an Always Branch has been added to a Page, whenever the script user prompts the system to move to the next logical page (by clicking a `Next` button, for example), the script user will *always* be brought to the Page specified in the branch statement.

Conditional

If a Conditional branch has been added to a Page, whenever the script user prompts the system to move to the next logical page (by clicking a `Next` button, for example), the script user may be brought to *different* Pages depending on the value of an Agent Scripting Field.

You would want to use a Conditional branch if a script user needs to go to a different Page depending on a customer's answer.

Return

If a branch of type `Return` has been added to a Page, whenever the script user prompts the system to move to the next logical Page (by clicking a `Next` button, for example), the script will *branch to* the Page defined in the most recent `Branch On Return` area of a `Branch Condition` dialog box.

Enabling the `Branch on Return` area and selecting a target Page is known as *defining a return point*. A return point can be defined on multiple Pages, but only the most recent definition is used.

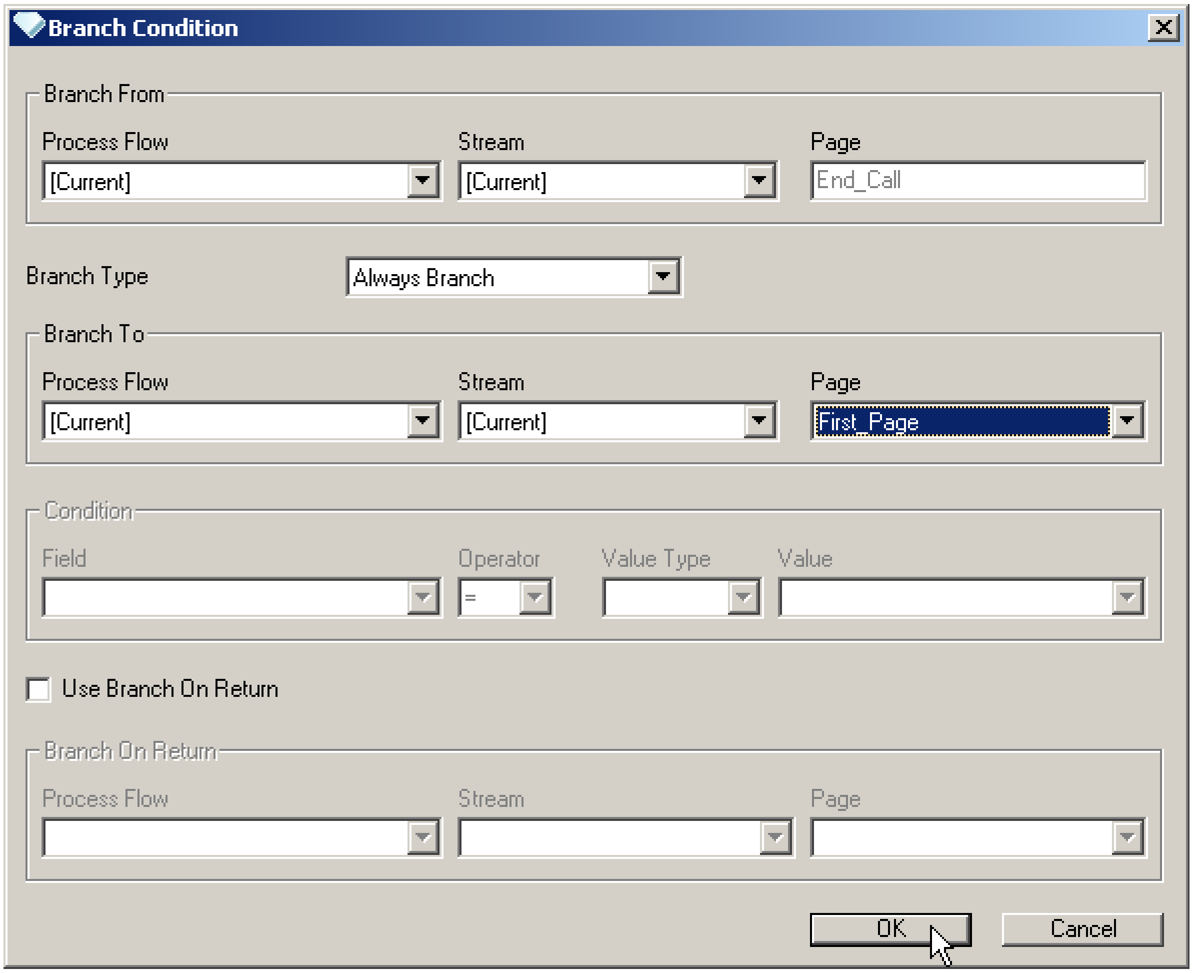
Stop

A Stop branch indicates that there is no logical next page. When a Stop branch is reached, it is as if the end of the Process Flow has been reached.

Creating Branches

While there are several methods for creating branches, let's do so using the `Page` menu.

- First use the `Page Tree` to open the Page for which you will create a branching statement.
- Select `Page` > `Insert Branch` to open the `Branch Condition` dialog box as shown in Figure 116.



Branch Condition

Branch From

Process Flow [Current]

Stream [Current]

Page End_Call

Branch Type Always Branch

Branch To

Process Flow [Current]

Stream [Current]

Page First_Page

Condition

Field

Operator =

Value Type

Value

□ Use Branch On Return

Branch On Return

Process Flow

Stream

Page

Figure 116:Branch Condition Dialog Box

Table 52describes the options on the `Branch Condition` dialog box.

Table 52: Branch Condition Dialog Box

Control	Description / What to Do...
Branch From:	This area defines the start point of the branch.
Process Flow	Select the Process Flow of the start point (such as `[Current]` for the current Process Flow).
Stream	Select the Stream of the start point (such as `[Current]` for the current Stream).

Table 52: Branch Condition Dialog Box (Continued)

Control	Description / What to Do...
Page	Assumes the current Page is the start point, and displays the current Page in a read-only field. If you want another start point, select a different Process Flow or Stream, or navigate to another Page within the same Process Flow and Stream.
Branch Type	Select the type of branch. Valid choices are `Always Branch, Conditional, Return,` and `Stop.`
Branch To:	This area defines the destination point of the branch.
Process Flow	Select the Process Flow of the destination point (such as `[Current]` for the current Process Flow).
Stream	Select the Stream of the destination point (such as `[Current]` for the current Stream).
Page	Select the destination Page from the `Page` dropdown list. This list displays all Pages that are part of the Process Flow and Stream selected in the adjacent fields. The default is the *first* Page if no Page is selected.
Condition:	(The following four fields are only available if the `Branch Type` is Conditional.) The Condition statement defines what condition must be met in order for the branch to execute.
Field	Select the Field from the `Field` dropdown list whose value you will use to set up the conditional statement. **Note:** Fields of type `Listener`, `String`, `Multiline Text`, `Numeric`, and `Date and Time` can be asynchronously updated using an Async Interface.

Table 52: Branch Condition Dialog Box (Continued)

Control	Description / What to Do...
Operator	Select the comparison operator from the Operator dropdown list. Genesys Agent Scripting automatically determines the valid operators and range of values based on the type of Field selected. For example, if you select a `Check Box` Field type, then only the equal (=) and not equal (<>) operators will be selectable, and only `True` and `False` for values will be selectable. Similarly, if you select a `Radio Button` Field type then only the equal (=) and not equal (<>) operators will be available. However, the values dropdown list displays all possible values for the radio button.
Value Type	Select a value type of `Field` if you want to compare the earlier Field's value to that of another Field. Select a value type of `Static` if you want to specify or select a static value for the conditional expression.
Value	The `Value` field is a dropdown list if you selected `Field` as a value type, or if you selected `Static` and the first comparison Field was a check box or radio button. The `Value` field is a text box if the value type is `Static` and the Field type was not a check box or radio button.
Use Branch On Return	This check box is only active if the branch type is `Always Branch` or `Conditional`. Selecting this check box enables you to define a return point. `Use Branch on Return` indicates that this branch is intended to be used in conjunction with a `Return` branch. When a branch that utilizes the `Branch on Return` feature is encountered, an entry is made on the Page navigation stack. When a `Return` branch is reached, the Page navigation stack is checked for any `Branch on Return` entries, and the most recent determines the Page that will be returned to. If no `Branch on Return` entries are found, an error will be indicated.

Table 52: Branch Condition Dialog Box (Continued)

Control	Description / What to Do...
Branch On Return:	(The following three fields are only available if Use Branch On Return is selected.) The Branch On Return statement defines what Process Flow, Stream, and Page to branch to when a return point is reached. **Note:** A return point must be defined on some Page in the Stream before the branch type Return is executed.
Process Flow	Select the Process Flow of the return point (such as [Current] for the current Process Flow).
Stream	Select the Stream of the return point (such as [Current] for the current Stream).
Page	Select the return point Page from the Page dropdown list. This list displays all Pages that are part of the Process Flow and Stream selected in the adjacent fields.
OK	Click OK to complete the branch statement and close the Branch Condition dialog box. Your new branch condition is visible in the Page Tree beneath the appropriate Page.

See "Navigation Actions" on page 114 of "Creating Actions" for an alternative way to create an Always Branch as an *Action* that can be placed as a button on a Page.

Examples

This section shows two step-by-step examples that illustrate how to create branches:

- Creating an Always Branch
- Creating Conditional Branches for varied customer responses

Setup

Carry out the instructions in the following subsections to prepare for the branching examples.

Procedure:
Creating the Pages

Start of procedure

1. Use Define > Pages to create a total of six new Pages with the following names:
   - First_Page
   - New_Service
   - Repair
   - Billing_Issue
   - Info_Request
   - End_Call
2. Add each Page to the current Stream using Stream > Insert Page.
3. Reorder the Pages in the Page Tree by clicking and dragging individual Pages through the list.

End of procedure

Create a Radio Button Field

- Refer to the chapter called "Creating Fields"and set up the Field shown in Table 53.

Table 53: Fields for Example

Field Label/Name	Field Type	Details
CustomerType	Radio Button	Select Alternate Format for Radio Button. Values are: • New Service • Repair • Billing Issue • Information Request

Create the Actions

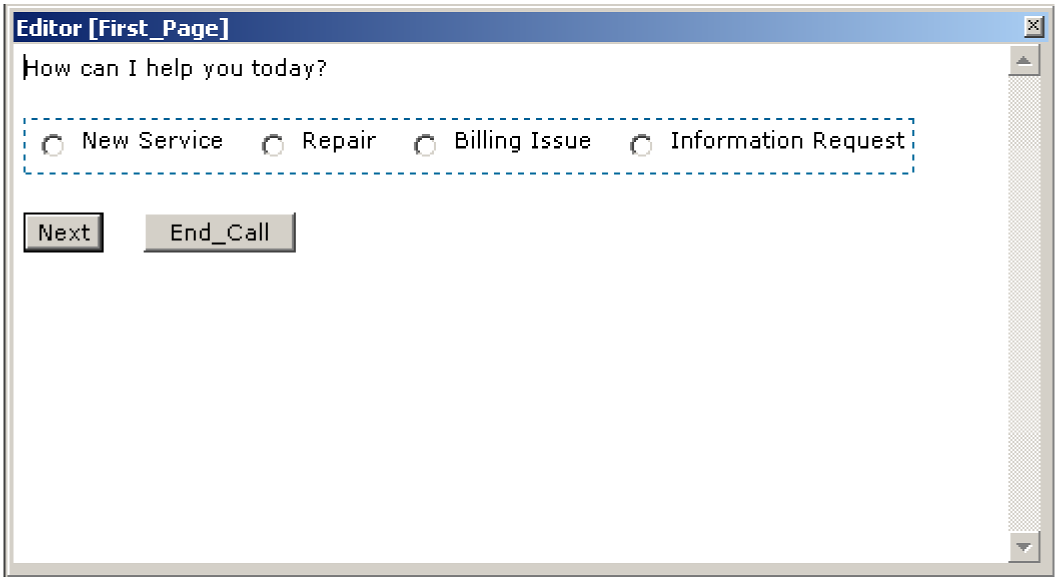
- Refer to the chapter called "Creating Actions" and set up the actions shown in Table 54.

Table 54: Actions for Example

Action Label/Name	Action Type	Details
Next	Next	Use Action wizard.
Previous	Previous	Use Action wizard.

End_Call	Branch	Use Action Wizard.
		Process Flow = [Current]
		Stream = [Current]
		Page = End_Call

Procedure:
Add Fields, Actions, and Text to the Pages



How can I help you today?

Figure 117:First_Page

Start of procedure

1. As shown in Figure 117, type How can I help you today? at the top of the First_Page.
2. Add the CustomerType field to the Page.
3. Add the Next and End_Call Actions to the Page.
4. Set up the other Pages as shown in Table 55.

Table 55: Text and Actions for Example

Page Name	What to Add

New_Service	Text = NEW SERVICE
	Action = End_Call

Repair	Text = REPAIR
	Action = End_Call

Billing Issue	Text = BILLING ISSUE
	Action = End_Call

Info_Request	Text = INFORMATION REQUEST
	Action = End_Call

End_Call	Text = END CALL
	Action = Next

End of procedure

Procedure:
Create an Always Branch

Purpose: Create an Always Branch so that when the script user reaches the End_Call Page, the script will always branch back to First_Page.

Start of procedure

1. Select the End_Call Page in the Page Tree.
2. Select Page > Insert Branch.
3. In the Branch Condition dialog box, select Always Branch from the Branch Type dropdown list.
4. In Branch To, select [Current] for Process Flow and Stream, and select First_Page in the Page dropdown list.
5. Click OK to complete the branch.

   When the script user clicks Next on the End_Call Page, the script will branch to First_Page.

   The End_Call Page in the Page Tree shows that an Always Branch has been set up, as shown in Figure 118.

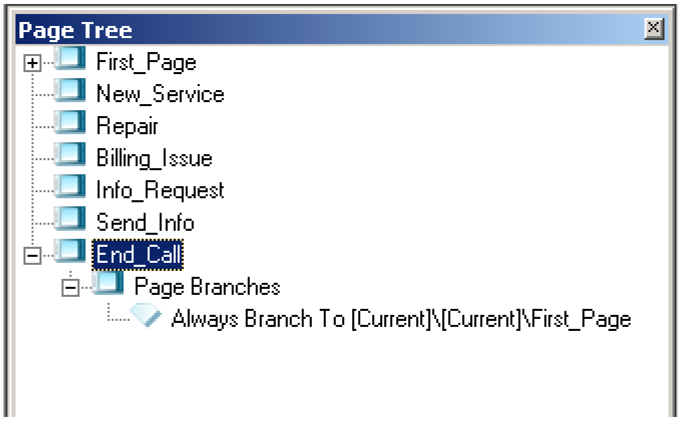

Page Tree
- First_Page
- New_Service
- Repair
- Billing_Issue
- Info_Request
- Send_Info
- End_Call
  - Page Branches
    - Always Branch To [Current]\[Current]\First_Page

Figure 118:End_Call Page Has Always Branch

End of procedure

Creating Conditional Branches

On First_Page we placed a Radio Button field to select the type of customer request. Let’s create Conditional branches so that the next script Page displayed will depend on the selection in this Field. For example, if the customer is calling for new service, branch to the New_Service Page; if there’s a repair question, branch to the Repair Page, and so on.

Procedure:
Create a Conditional Branch

Start of procedure

1. Select the First_Page Page in the Page Tree.
2. Select Page > Insert Branch.
3. In the Branch Condition dialog box, select Conditional from the Branch Type dropdown list.
4. In Branch To, select [Current] for Process Flow and Stream, and select New_Service from the Page dropdown list.
5. In Condition, select CustomerType in the Field dropdown list.
6. Select the equals (=) Operator.
7. Select Static from the Value Type dropdown list.
8. Select New Service from the Value dropdown list.

   The preceding steps have created a conditional expression that can be interpreted as follows:

"Branch to the New_Service Page if the value of the CustomerType Radio Button field is New Service."

9. Click OK to complete the branch.

10. Repeat Steps 2 through 8 above, but substitute your values for Branch To... Page and Condition... Value as shown in Table 56.

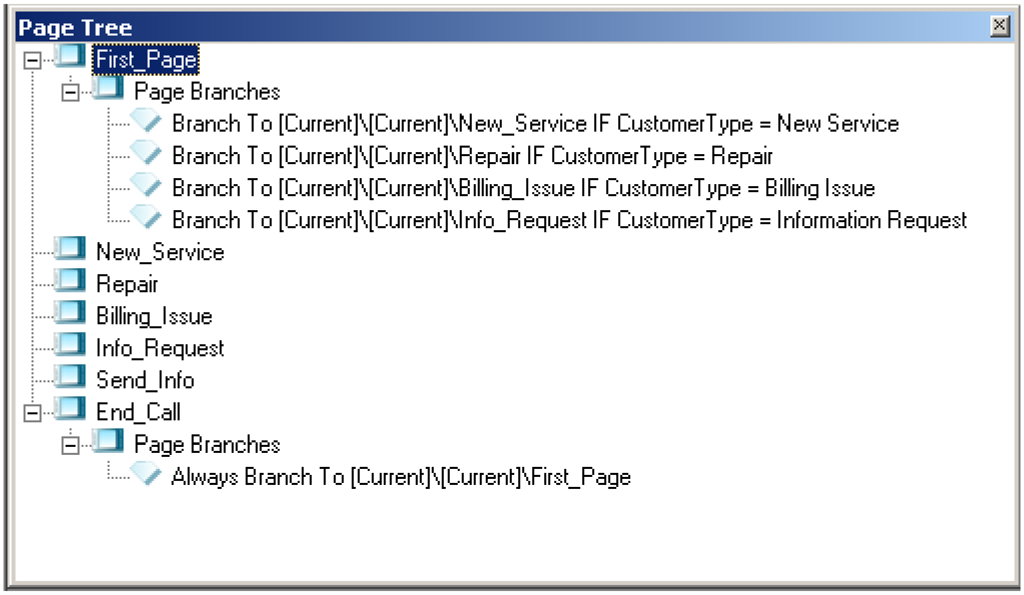
Table 56: Text and Actions for Example

Branch To Page	Condition Value
Repair	Repair
Billing Issue	Billing Issue
Info_Request	Information Request

11. Click OK to complete each branch.

When the script user clicks Next on First_Page, the script will branch to different Pages depending on what was selected in the Radio Button field.

The First_Page Page in the Page Tree shows that Conditional branches have been set up, as shown in Figure 119.



Page Tree

```
First_Page
    Page Branches
        Branch To [Current]\[Current]\New_Service IF CustomerType = New Service
        Branch To [Current]\[Current]\Repair IF CustomerType = Repair
        Branch To [Current]\[Current]\Billing_Issue IF CustomerType = Billing Issue
        Branch To [Current]\[Current]\Info_Request IF CustomerType = Information Request
New_Service
Repair
Billing_Issue
Info_Request
Send_Info
End_Call
    Page Branches
        Always Branch To [Current]\[Current]\First_Page
```

Figure 119:First_Page Page Has Conditional Branches

End of procedure

Compile and Test

When you have a script segment that you wish to test, such as the example we've just completed above, you should compile the code and run `Simulate` to see how things work.

The procedures for compiling and testing (simulating) your code are described in Chapter 12, "Compiling, Testing, and Deploying Your Application," on page 269.

Chapter

12 Compiling, Testing, and Deploying Your Application

This chapter provides an overview of the process of compiling scripts into ASP, ASPX, or JSP, testing the application by simulating what a script user sees, and deploying the compiled and tested application to the script user population.

The information in this chapter is divided among the following topics:

What Does it Mean to Compile, Test, and Deploy?

The Genesys Agent Scripting Development Environment allows you to build sophisticated and powerful scripting applications. You must *compile* the code you generate into one of the following code targets:

- Active Server Pages (ASP) if the application will run on a Microsoft IIS web server.
- Active Server Pages - Extended (ASPX) if the application will run on a Microsoft IIS web server utilizing the .NET Framework v1.1 support.
- Java Server Pages (JSP) if the application will run on a Java web server such as Apache Tomcat.

Compiling is the process Agent Scripting uses to convert Project Books into pages (ASP, ASPX, or JSP) that can be used in the production environment.

Once the code is compiled, you can *test* the functionality of the application to assure that it works as intended. You do this using the Agent Scripting `Simulate` function, which lets you see and work with the application as a script user would. Compatibility with `JMeter` allows you to simulate one-to-many script users and thereby judge how your application stands up in a rigorous environment where more throughput is required.

Testing often reveals areas of the application that need to be corrected, enhanced, or modified in some way. After making changes and improvements, you can compile and test again to see the results of your changes immediately.

Once the application is production-ready, you *deploy* the application *for production* so that it is made available to the population of script users for whom it is intended.

Note: When specifying a `Target URL` for your Target Environment, use the full computer name or IP address (http://servername or http://xxx.xxx.xxx.xxx). Do *not* use http://localhost.

Creating a Target Environment

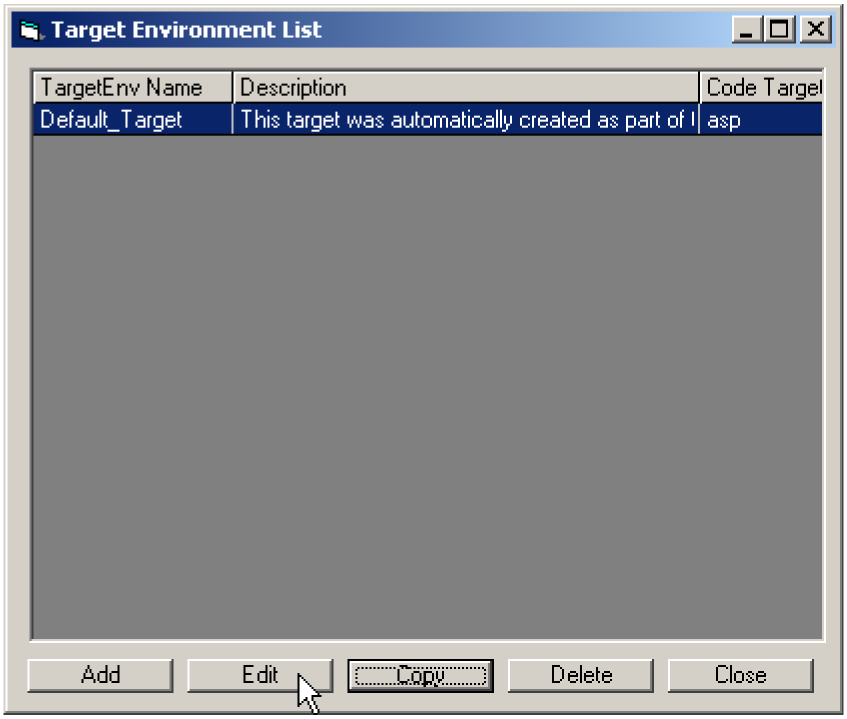
The Target Environment specifies the directory to which the code will be compiled, the type of code to generate, and other necessary information about your application's development environment.

Procedure: Create or Select a Target Environment

Purpose: Create or select a Target Environment using the `Compile` menu.

Start of procedure

1. Select Compile > Target Environment to open the Target Environment List dialog box as shown in Figure 120.



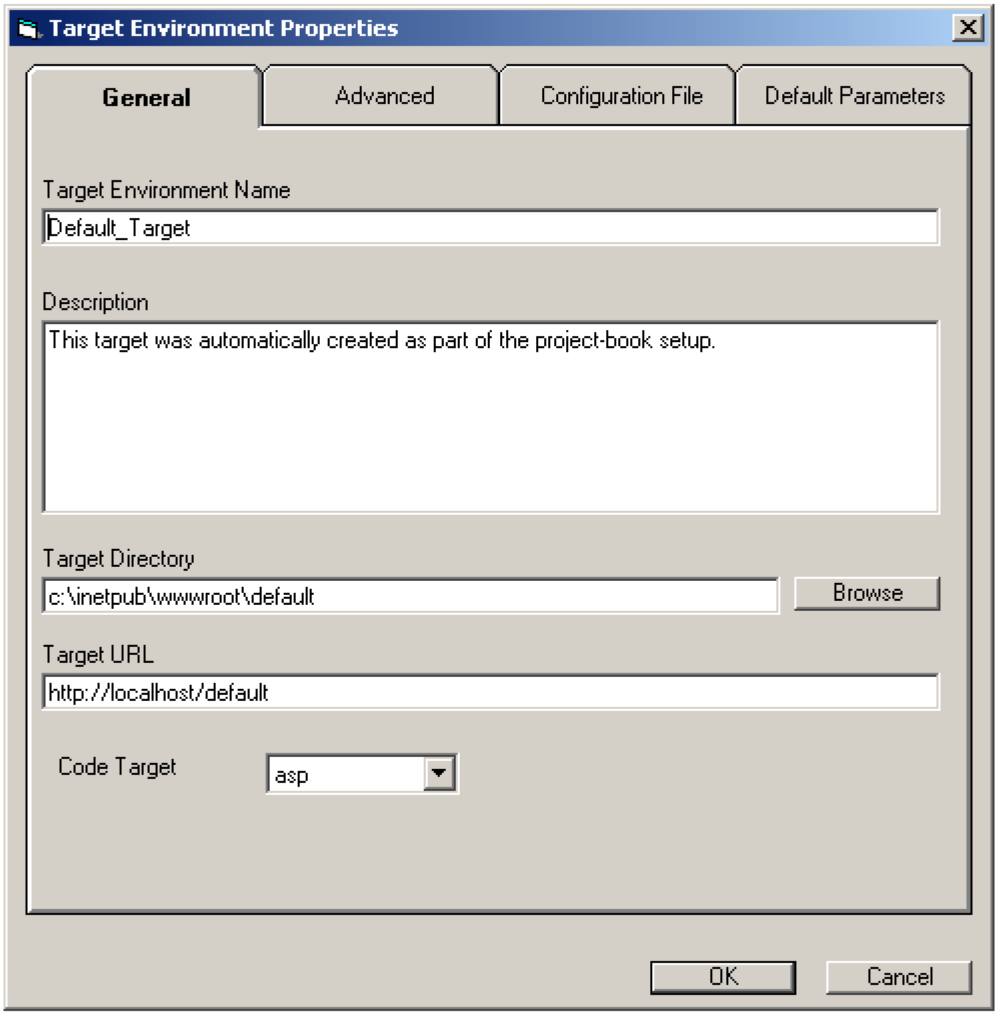
TargetEnv Name	Description	Code Targe
Default_Target	This target was automatically created as part of	asp

Figure 120:Target Environment List Dialog Box

When you created your new Project Book as described in the “Creating a New Project Book” chapter, you set up a *default* Target Environment which should already appear in the Target Environment List.

2. Click Edit.

The default Target Environment may resemble (if you selected ASP) what is shown in Figure 121.



Target Environment Name: Default_Target

Description: This target was automatically created as part of the project-book setup.

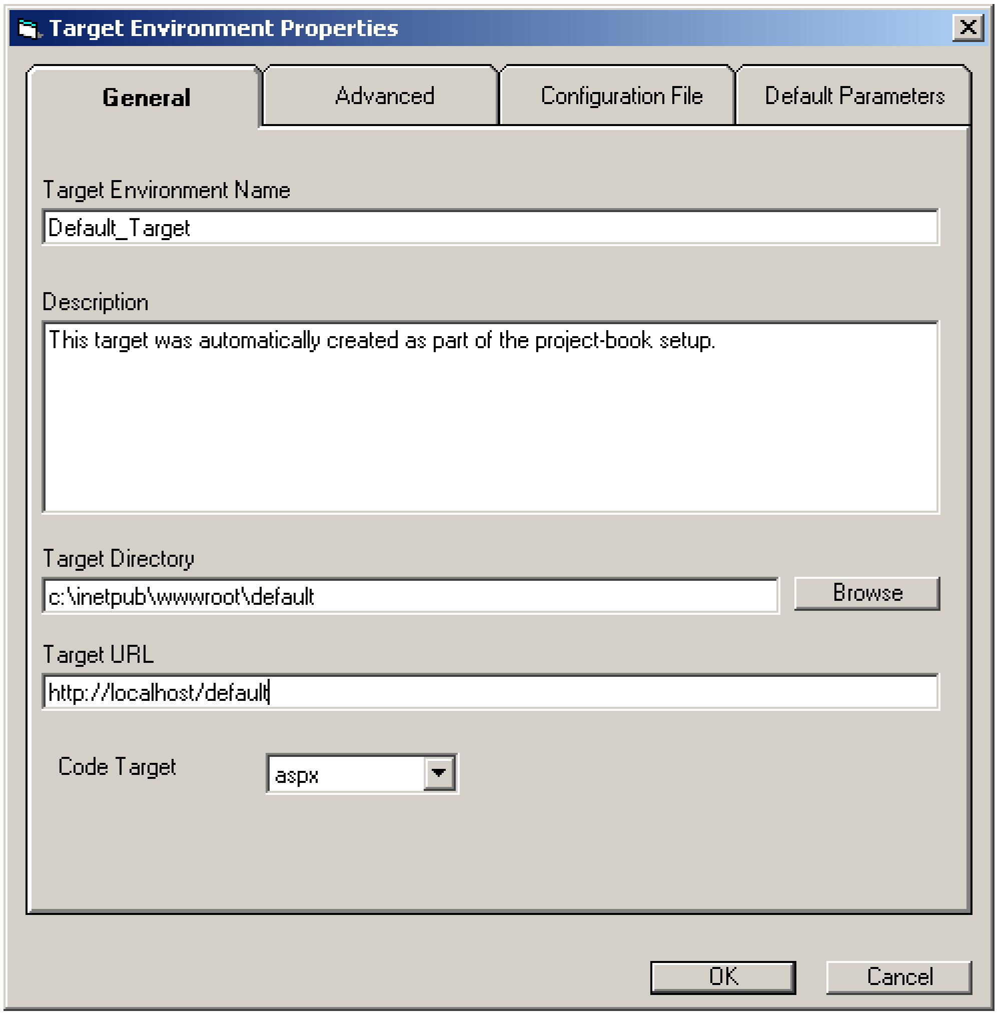
Target Directory: c:\inetpub\wwwroot\default

Target URL: http://localhost/default

Code Target: asp

Figure 121:Target Environment Properties Dialog Box - ASP Example

Or, the default Target Environment may resemble (if you selected ASPX) what is shown in Figure 122



Target Environment Name: Default_Target

Description: This target was automatically created as part of the project-book setup.

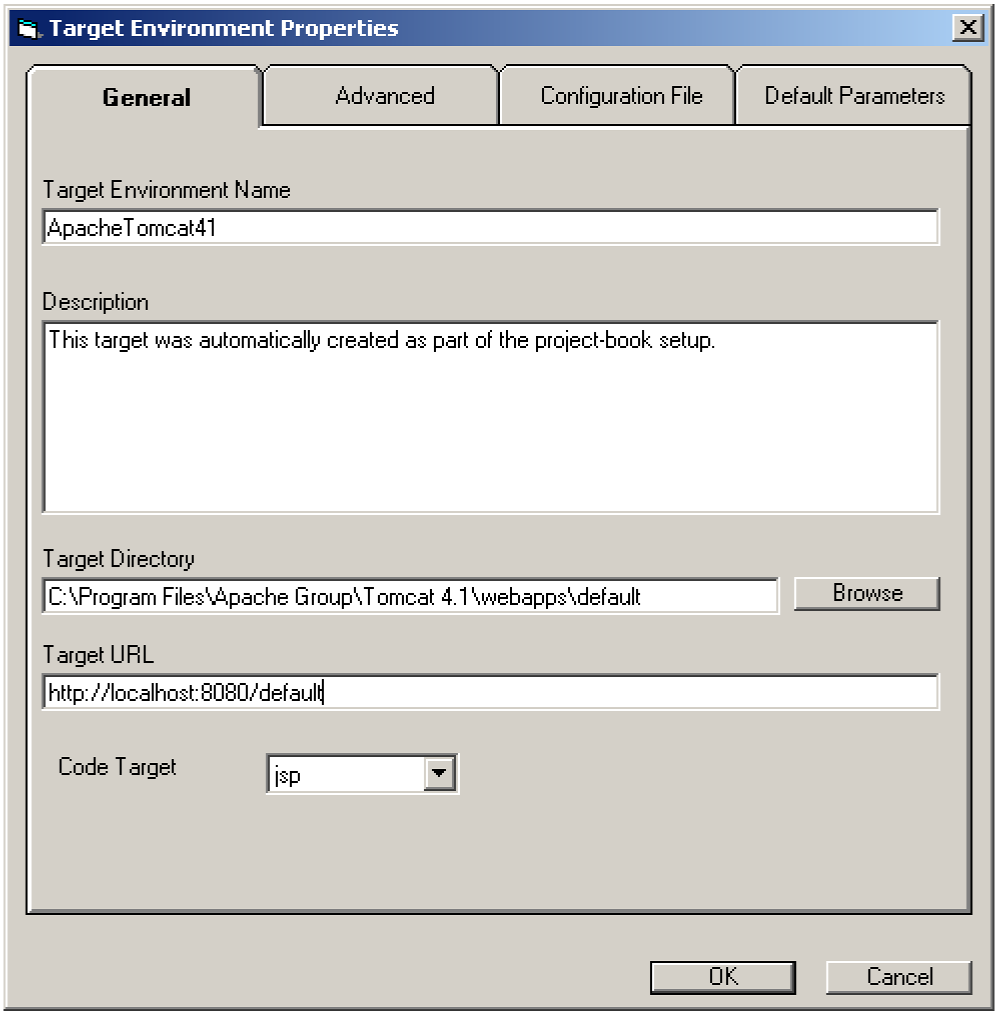
Target Directory: c:\inetpub\wwwroot\default

Target URL: http://localhost/default

Code Target: aspx

Figure 122:Target Environment Properties Dialog Box - ASPX Example

Finally, the default Target Environment may resemble (if you selected JSP) what is shown in Figure 123.



Target Environment Name: ApacheTomcat41

Description: This target was automatically created as part of the project-book setup.

Target Directory: C:\Program Files\Apache Group\Tomcat 4.1\webapps\default

Target URL: http://localhost:8080/default

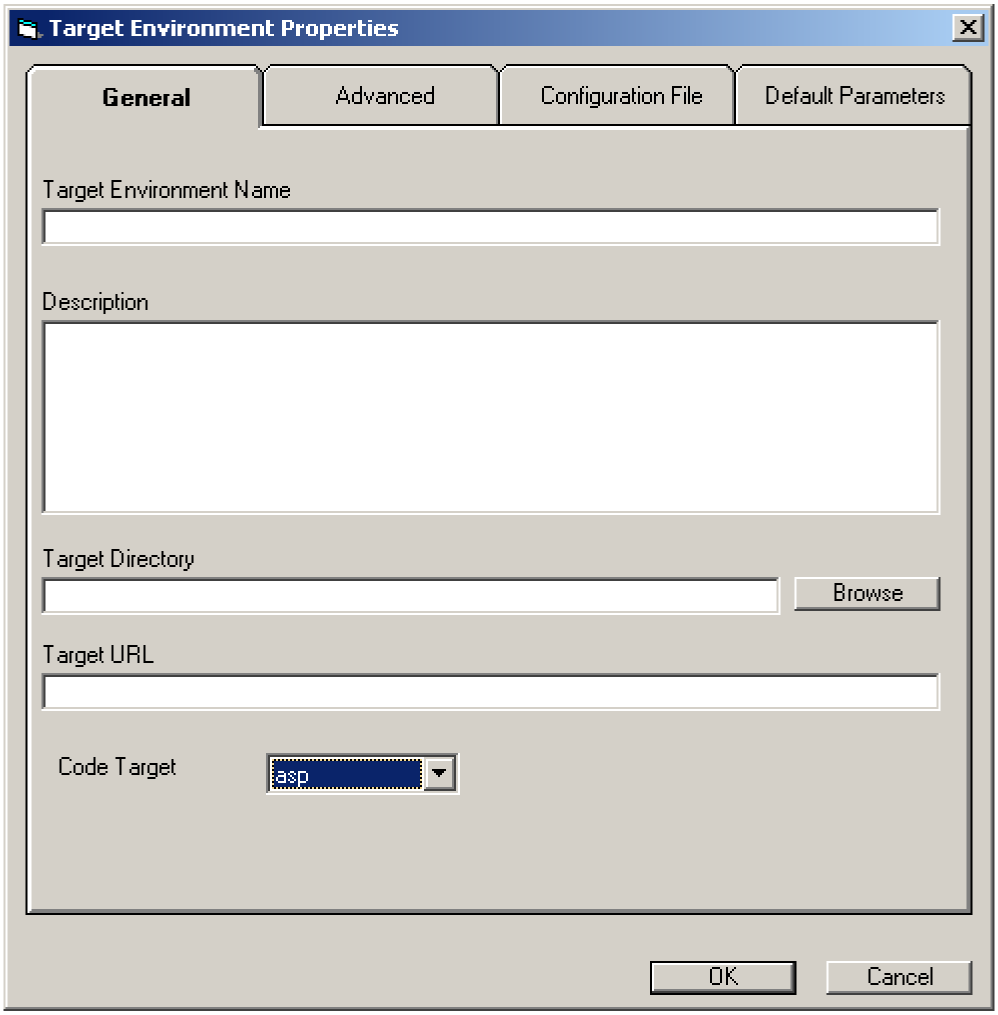
Code Target: jsp

Figure 123:Target Environment Properties Dialog Box - JSP Example

The `Target Environment Properties` dialog box is where you define the characteristics that are unique to each Target Environment you wish to create.

3. Click `OK` in the `Target Environment Properties` dialog box after reviewing the properties for the default environment.
4. Click `Add` in the `Target Environment List` to initiate the process for creating a new Target Environment.

   An empty `Target Environment Properties` dialog box opens as shown in Figure 124.



Code Target asp

Figure 124:New Target Environment Properties Dialog Box - General Tab

5. Click OK to save your changes or Cancel to abandon them.

End of procedure

General Tab

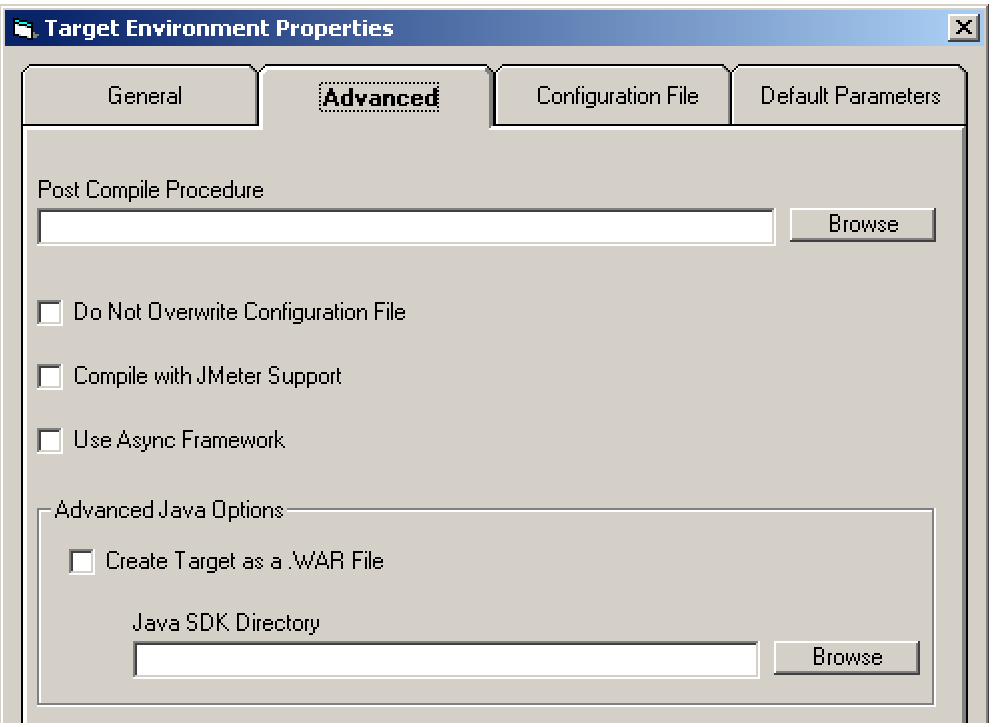
Table 57 describes the options on the `General` tab of `Target Environment Properties`.

Table 57: Target Environment Properties, General Tab

Control	Description / What to Do...
Target Environment Name	The Target Environment Name identifies the Target Environment. Type a name for the Target Environment you are creating.
Description	(Optional) This field contains a description of the Target Environment. It is used for informational purposes and is not used elsewhere in the system.
Target Directory	The Target Directory is used to indicate a folder for Genesys Agent Scripting to place the completed, generated scripts. This location can be a folder on the machine on which Genesys Agent Scripting is installed or a folder on a network drive. The only restriction is that you must have write access to this location. Example for ASP or ASPX: c:\inetpub\wwwroot\default Example for JSP: C:\Program Files\Apache Group\Tomcat 4.1\ webapps\default
Browse	Click Browse to navigate to a target directory that you wish to select for generated scripts.
Target URL	The Target URL is the location from where the Genesys Agent Scripting Web Server pages are accessed. It is used for running Simulations and Rules Assignment.
Code Target	The Code Target dropdown list allows you to select the platform for the application. Currently, Genesys Agent Scripting can produce pages as JSP (Java Server Pages) for running on Java-based web platforms, or ASP (Active Server Pages) and ASPX (ASP.NET) for running on IIS.
OK	Click OK to complete setup of your new Target Environment and close the Target Environment Properties dialog box.

Advanced Tab

Figure 125shows the `Advanced` tab of the `Target Environment Properties` dialog box.



Post Compile Procedure

□ Do Not Overwrite Configuration File

□ Compile with JMeter Support

□ Use Async Framework

Advanced Java Options

□ Create Target as a .WAR File

Java SDK Directory

Figure 125:Target Environment Properties Dialog Box - Advanced Tab

Table 58 describes the options on the `Advanced` tab of `Target Environment Properties.`

Table 58: Target Environment Properties, Advanced Tab

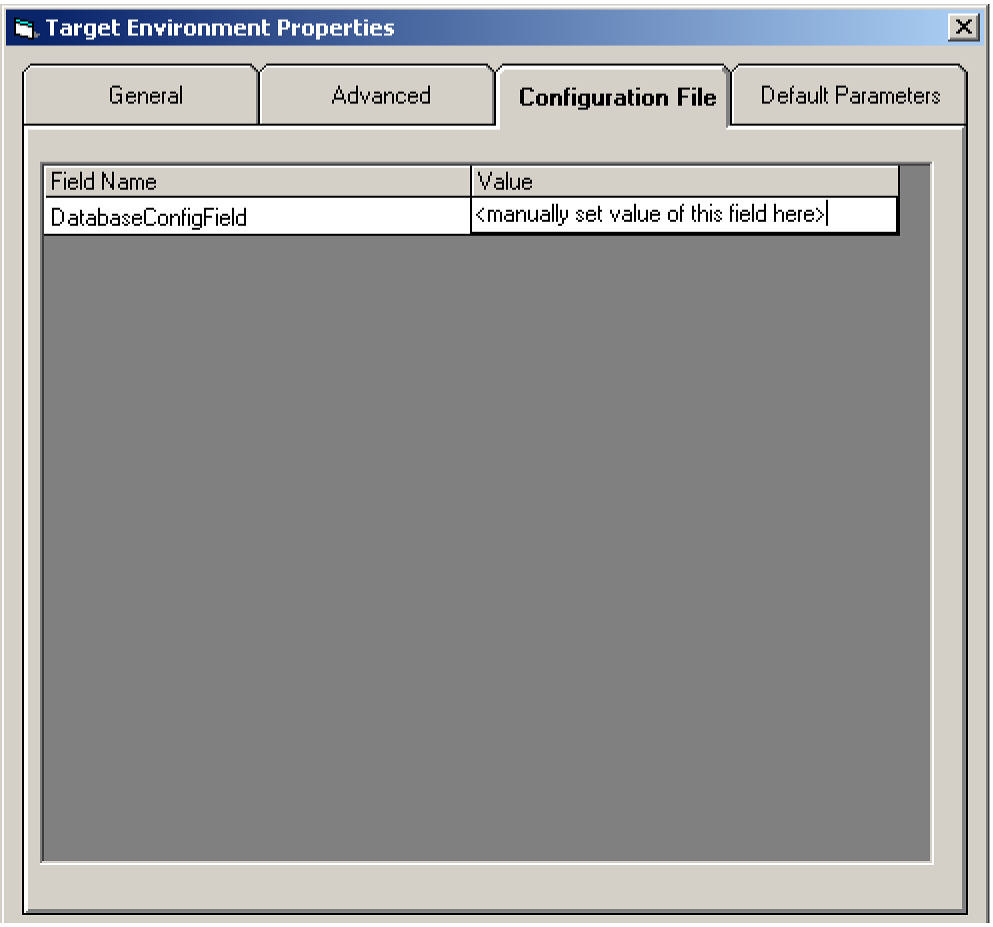
Control	Description / What to Do...
Post Compile Procedure	A *post compile procedure* allows you to run a batch file as soon as Genesys Agent Scripting has finished compiling the scripts. Type the full path of the file to be run, or click `Browse` to navigate to the file. An example of a post compile procedure is a script that uploads the compiled scripts to the production web server.
Browse	Click `Browse` to navigate to a post compile procedure batch file.

Table 58: Target Environment Properties, Advanced Tab (Continued)

Control	Description / What to Do...
Do Not Overwrite Configuration File	If this check box is selected, the `Configuration File` tab is disabled. This prevents you from assigning values to configuration file Fields in the `Configuration File` tab, which will overwrite existing values in the configuration file.
Compile with JMeter Support	If this option is enabled, it allows recording of the JMeter scenario for the current session. This is useful for testing the throughput of a Genesys Agent Scripting Web Server. A generated scenario is available at `<Target URL>/WWGURLView.jsp`. This page presents an HTTP request to the Genesys Agent Scripting Web Server, and it tests the elements in XML format ready for a copy-paste into the JMeter test project. The recorded script is saved as a JMeter Workbench in the `<Target Directory>/WWG.jmx` file. Generated scripts do not specify the Genesys Agent Scripting Web Server host and port and can be reused on multiple environments. **Note:** To have the application create a log for debugging purposes, select `Compile` > `Trace Options` and set the Trace Level.
Use Async Framework	This option is only enabled for code targets of ASPX and JSP. If selected, it enables the use of the asynchronous events feature.
Advanced Java Options	The following options are enabled for a code target of JSP only.
Create Target as a .WAR File	Select this check box to generate a `.WAR` file in the target directory that can be used for deployment to Application Servers (Tomcat and WebSphere).
Java SDK Directory	Provide the location of the utility from the Java SDK that is used to create the `.WAR` file, or click `Browse` to navigate to the directory that contains this utility. **Note:** If you do not provide the path here, then you must either set an environment variable called `JAVA_HOME`, or place the Java SDK in the host machine's `PATH`.
Browse	Click `Browse` to navigate to the directory that contains the Java SDK utility used to create `.WAR` files.

Configuration File Tab

Figure 126shows the `Configuration File` tab of the `Target Environment Properties` dialog box.



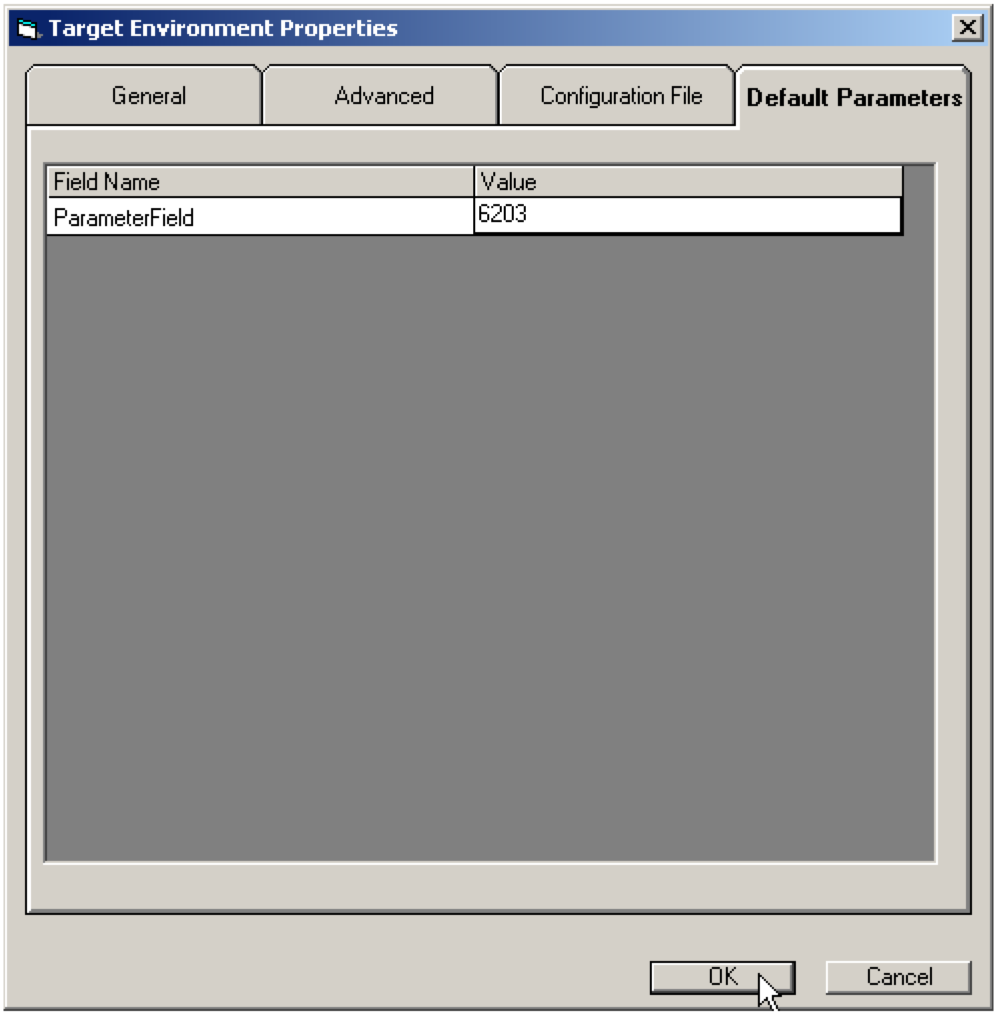
Field Name	Value
DatabaseConfigField	<manually set value of this field here>

Figure 126:Target Environment Properties Dialog Box - Configuration File

A *configuration file* is an editable, external file (`WWGConfig.xml`) whose data is passed to the application at startup. The `Configuration File` tab contains a list of all Fields that have been marked as configuration file Fields. You can change the value of the Field without having to open the configuration file and edit it directly.

Default Parameters Tab

Figure 127shows the `Default Parameters` tab of the `Target Environment Properties` dialog box.



Field Name	Value
ParameterField	6203

Figure 127:Target Environment Properties Dialog Box - Default Parameters

A *parameter* is a Field whose value is passed in by another system. The `Default Parameters` tab contains a list of all Fields that have been marked as parameters. You can set a default value for the Field. When simulated, the parameters will appear on the start screen with the values set here.

The values that are passed in by another system are passed via the URL. For example:

```
http://<servername> or <IP 
address>/default/WWGStart.asp?WWGProcessFlowName=myFlow&parm1=123&parm2
=abc
```

where `parm1` and `parm2` are Agent Scripting Fields that were configured as parameters.

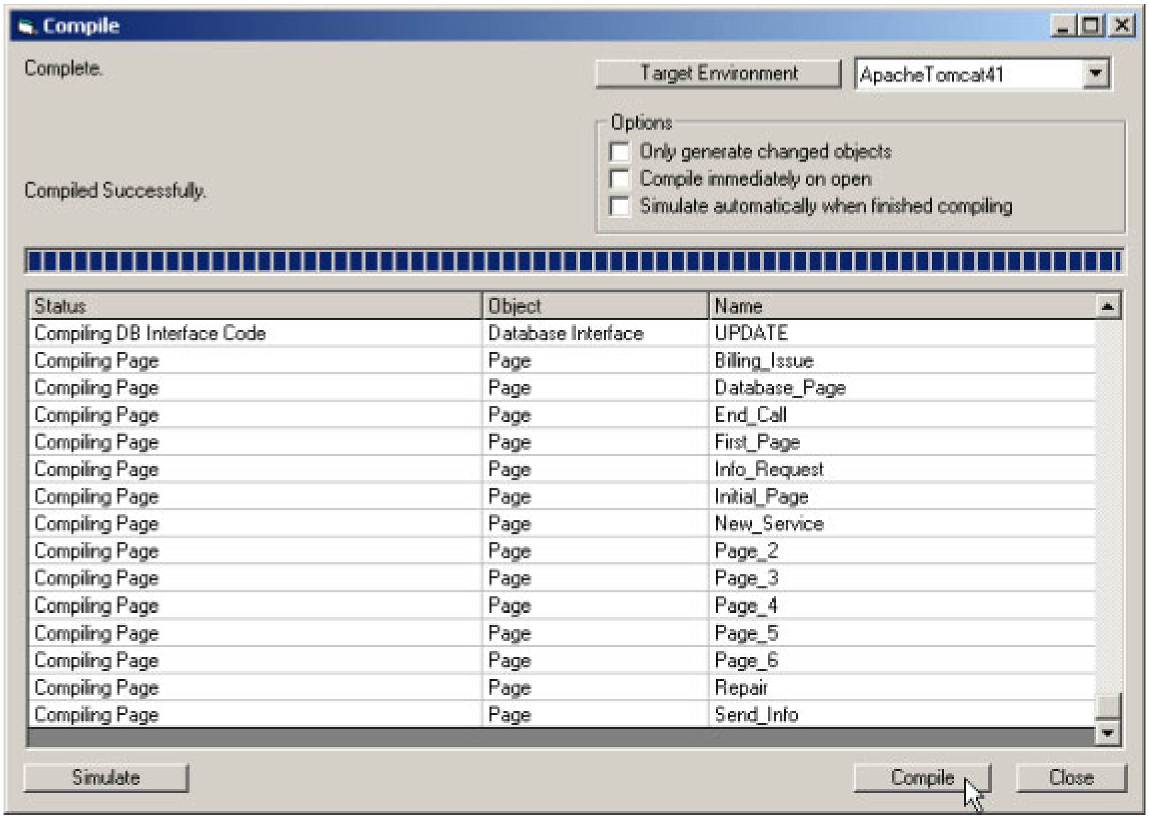
Compiling a Script

After you have created pages and are ready to test them, you must compile. This option takes the Pages that you created and produces the Genesys Agent Scripting compiled pages as ASP, ASPX, or JSP. The `Compile` option compiles all objects (for example, Pages, Fields, actions) in the current Project Book. If there are a large number of Pages, this might take some time. In order to save time, you can choose to compile *incremental* changes only.

Notes: Before compiling a script, be sure that you have set up an appropriate Target Environment as described in "Creating a Target Environment" on page 270. You need a viable `Target Directory` string in order for Genesys Agent Scripting to compile your code successfully.

ASP scripts using the Agent Scripting Toolkit do not support Suspend and Resume actions, and they will return XML Interface warnings when compiled.

Select `Compile` > `Compile` to open the `Compile` dialog box (see Figure 128).



Complete.

Compiled Successfully.

ApacheTomcat41

□ Only generate changed objects
□ Compile immediately on open
□ Simulate automatically when finished compiling

Status	Object	Name
Compiling DB Interface Code	Database Interface	UPDATE
Compiling Page	Page	Billing_Issue
Compiling Page	Page	Database_Page
Compiling Page	Page	End_Call
Compiling Page	Page	First_Page
Compiling Page	Page	Info_Request
Compiling Page	Page	Initial_Page
Compiling Page	Page	New_Service
Compiling Page	Page	Page_2
Compiling Page	Page	Page_3
Compiling Page	Page	Page_4
Compiling Page	Page	Page_5
Compiling Page	Page	Page_6
Compiling Page	Page	Repair
Compiling Page	Page	Send_Info

Figure 128:Compile Dialog Box

Note: Beginning with release 7.2.0, all options set in the Compile dialog box are specific to the Target Environment. When you change the Target Environment, the editor displays the options that are specific to your new choice.

Table 59 describes the options on the Compile dialog box.

Table 59: Compile Dialog Box

Control	Description / What to Do...
Target Environment	Click the Target Environment button to select the Target Environment you want to edit. The corresponding dropdown list allows you to select from all defined Target Environments.
Options:	Three check boxes provide options that you can use to tailor your compile operation.
Only Generate Changed Objects	Select this check box to have Genesys Agent Scripting only compile objects that have been changed since the last compile. This cuts down on compile time for large Project Books.
Compile Immediately on Open	Select this check box to cause Genesys Agent Scripting to compile the code automatically whenever Compile > Compile is selected. This setting takes effect the next time Compile is selected.
Simulate Automatically When Finished Compiling	Select this check box if you want the simulation to be automatically launched in a web browser as soon as Genesys Agent Scripting has finished compiling.
Status/Object/Name	This area of the dialog box displays information during the compile operations about which objects and object types are being compiled.
Simulate	After compiling is complete, you click Simulate to launch the simulation as described in “Testing the Compiled Code with Simulate” on page 283. If Simulate Automatically When Finished Compiling is selected, you will not need to click Simulate.
Compile	Click Compile to convert the objects in the Project Book into ASP, ASPX, or JSP code.
Close	Click Close to close the Compile dialog box.

Testing the Compiled Code with Simulate

After you have finished editing the scripts and have compiled them, they are ready to be tested.

Once the scripts have been deployed, you can invoke the `Simulate` operation. If the script designer has write access to a web server's application directory, the target directory can be set to the web server, and the `Compile` operation will load the scripts into the web server, ready for execution. Otherwise, see your web server administrator to deploy the scripts.

Notes: Before running `Simulate,` be sure to *start the web server* for the appropriate code type (Microsoft IIS for ASP or ASPX, or Apache Tomcat for JSP). Furthermore, be sure the web server has access to the Target Directory path. For JSP, for example, log in to the Tomcat Manager (`http://<servername> or <IP address>:8080/manager/html`) to add the directory path for the application. For ASP or ASPX, go to `Internet Information Services` under `Administrative Tools` in Windows, find the Target Directory, open `Properties,` and click `Create` to add the *generated scripts* as an application for this directory. Be sure to set `Write` permissions to the directory as well. See "Deployment Notes" on page 287.

Notes: Apache Tomcat 5.0 requires that two additional components (.jar files) be added to the `\common\lib` or `\shared\lib` folders within the directory path where Tomcat 5.0 is installed:

`mail.jar`

`activation.jar`

These are available from the most current version of JavaMail and Java Beans Activation Framework available on the Sun website.

Run `Simulate` in one of the following ways:

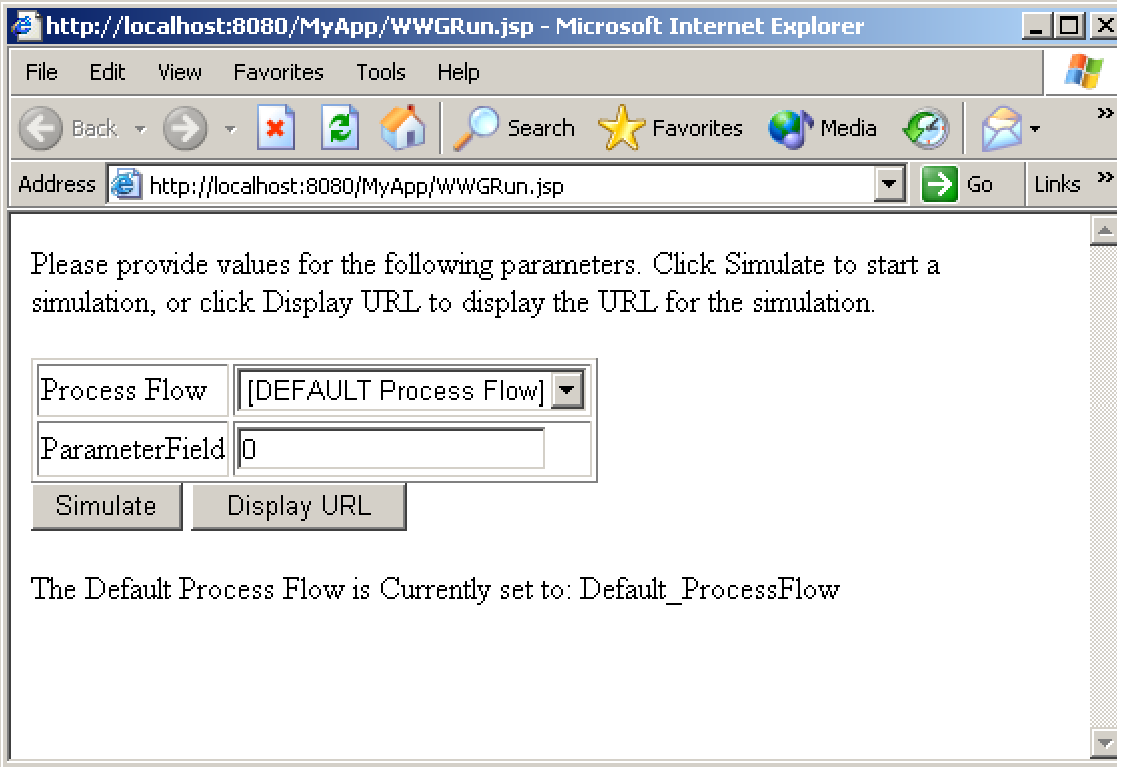
- Select `Compile` > `Simulate,` or

  If the `Compile` dialog box is open, click the `Simulate` button, or

  Select `Compile` > `Compile` to open the `Compile` dialog box, then click the `Simulate` button.

In all cases, you must have already compiled the code and, as stated, your web server must be running.

A web browser window opens to the `Script Launch Page,` as shown in Figure 129.


http://localhost:8080/MyApp/WWGRun.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/MyApp/WWGRun.jsp Go Links

Please provide values for the following parameters. Click Simulate to start a simulation, or click Display URL to display the URL for the simulation.

Process Flow	[DEFAULT Process Flow]
ParameterField	0

The Default Process Flow is Currently set to: Default_ProcessFlow

Figure 129:Script Launch Page

Table 60 describes the controls on the `Script Launch Page`.

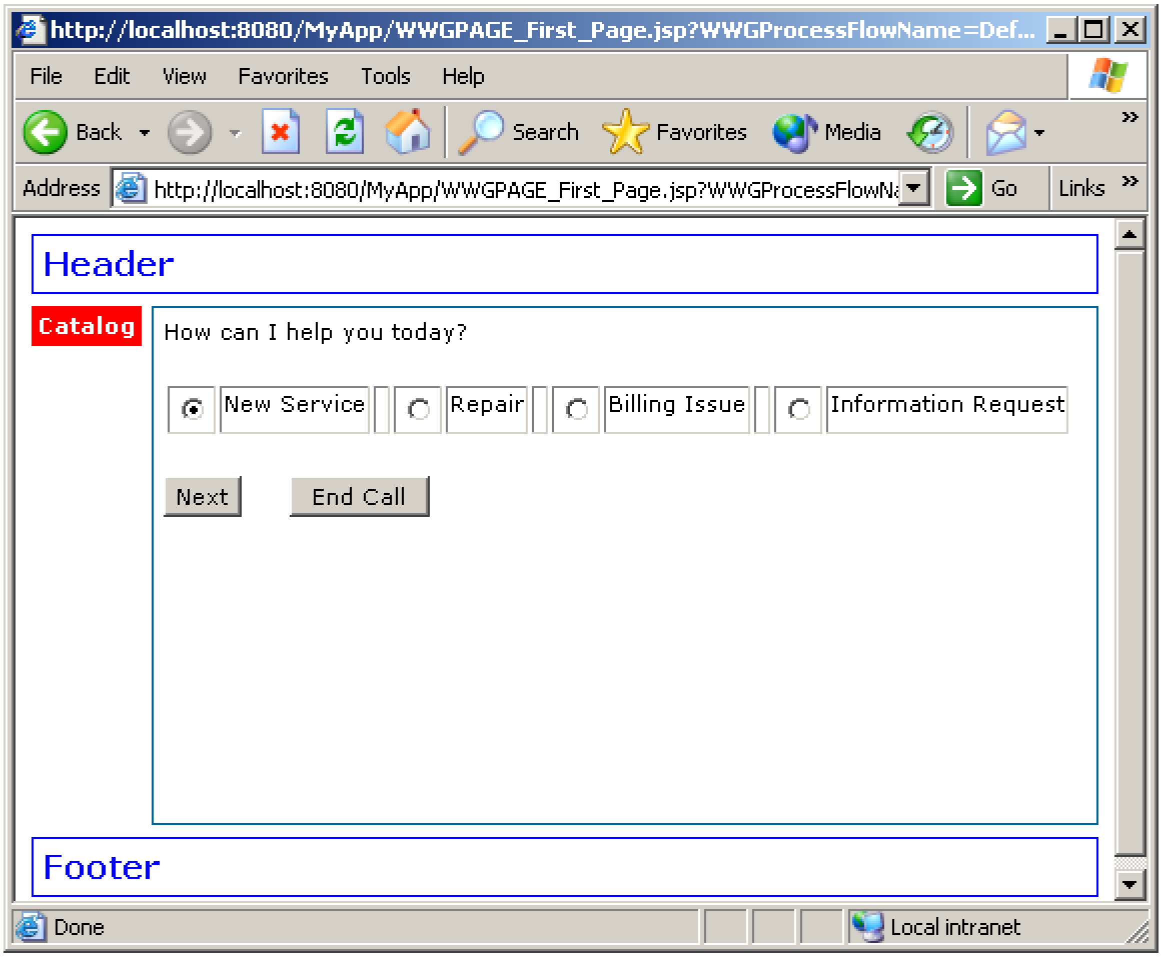
Table 60: Script Launch Page

Control	Description / What to Do...
Process Flow	Select a script for Genesys Agent Scripting to run. This dropdown list contains a list of existing scripts as well as a `[Default Process Flow]` entry as the first entry in the list. If you select the `[Default Process Flow]` entry, then Genesys Agent Scripting automatically uses the first Page of the first Stream as the default Page.
Parameters	All Fields defined as parameters appear on this Page. The example in Figure 129 on page 284 shows one Field called `ParameterField`. Any Field that is defined as type `Dropdown List` or `Radio Button` will appear as a dropdown list. All other Fields will appear as text fields. You may assign values to these Fields before running the Genesys Agent Scripting application in simulation mode.

Table 60: Script Launch Page (Continued)

Control	Description / What to Do...
Simulate	After you have supplied values for the parameters and selected a Process Flow, click `Simulate`. You will be placed in the first Page of the selected script. You can navigate through the Pages and test them before setting up the Genesys Agent Scripting Application for general use.
Display URL	Click the `Display URL` button to cause the starting URL, an example of which is seen below, to appear in the browser window. Example Starting URL: `http://<servername> or <IP address>:8080/default/WWGStart.jsp?WWGProcessFlowName= Default_ProcessFlow&ParameterField=0`

Figure 130shows an example of the first Page of a script running in `Simulate` mode.


http://localhost:8080/MyApp/WWGPAGE_First_Page.jsp?WWGProcessFlowName=Def...
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Media
Address
http://localhost:8080/MyApp/WWGPAGE_First_Page.jsp?WWGProcessFlowN:
Go
Links
Header
Catalog
Next
End Call
Footer
Done
Local intranet

How can I help you today?

● New Service ○ Repair ○ Billing Issue ○ Information Request

Figure 130:Script Running in Simulate Mode

Deploying the Application

The steps required to deploy a script can vary greatly based on a number of factors, including Target Environment, web server, network administration policies, and security policies, to name a few. The steps can range from simply setting the target directory of the `Compile` step appropriately, to complying with a number of business standards that adhere to company policies regarding web applications, which goes beyond the scope of Genesys Agent Scripting. Genesys Agent Scripting does provide a *Post Compile Procedure* option to assist the application developer in implementing these requirements.

Minimum Requirements

The minimum requirements for deploying a Genesys Agent Scripting application are:

- The Compile operation must be done with correct URL for the production web server.
- All files from the target directory must be placed in an application directory for the web server.
- The web server must be able to identify the application directory as a web application.
- The web server must have write authority to the application directory.

Deployment Steps

After you have completed the following steps:

- Developed the script
- Thoroughly tested the script's functionality and operation
- Made the necessary changes to the script
- Retested functionality
- Confirmed that the application is ready for use

then it is time to deploy your application.

Deployment of the application is not much different than the steps you've already carried out, although the function itself may be carried out by another individual or department (such as a system administrator or the IT or MIS department).

These are the general steps involved in deploying an application that has been thoroughly tested and is ready to go:

- Identify where on the appropriate computer the application will reside.
- This could be a directory path on a networked server that the script user population will have read access to through a web browser.
- Set up a new Target Environment within the Genesys Agent Scripting Development Environment that identifies that server path as the Target Directory, and select the code target (ASP, ASPX, or JSP).
- Compile the script using the Target Environment you just set up.
- This should upload all necessary files to the destination path.
- Test that the application runs by opening a web browser (preferably on a computer networked to the server that has the application), and then type the application URL in the browser's Address box.
- Run through the application again to check for any last-minute issues.
- Provide the application URL to the script user population so that the target audience can access and run the application.

Deployment Notes

You should consider the following deployment issues.

Microsoft IIS (ASP, ASPX) Deployment

You must create an *application* in IIS Manager.

Procedure:
Create an *Application* in IIS Manager

Start of procedure

1. In the Windows Control Panel, open `Administrative Tools` and double-click `Internet Information Services`.
2. Expand the tree to `Web Sites` > `Default Web Site` > *[yourApplication]*, right-click and select `Properties`.
3. Click the `Create` button beside `Application Name`. You must do this in order for IIS to know that this a runnable application. If you don't, you will get an "Application Setup Error" when starting the application.
4. Give write permission to the folder where you have deployed the application. If you don't, the logging function will fail and display an error message:

   ```
   The page cannot be displayed.
   HTTP 500.100 - Internal Server Error - ASP error
   Internet Information Services
   ```

End of procedure

Procedure:
Deploy ASPX on a Windows 2003 computer

Start of procedure

1. Launch Windows Explorer.
2. Navigate to your application directory—for example, `c:\inetpub\wwwroot\<dirname>`.
3. Right-click your application directory and select `Properties`.
4. In the `Properties` dialog box, click the `Security` tab.
5. In the `Group or user names` window, select `IIS_WPG`.

   If `IIS_WPG` is not presented in this window:

   a. Click the `Add` button to open the `Select Users, Computers, or Groups` dialog box.

   b. In the `Enter the object names to select` window, type `IIS`, and then click the `Check Names` button. `IIS_WPG` will appear.

c. Click OK to return to the Properties dialog box, where you will now see IIS_WPG in the Group or user names window.

6. In the Permissions window, under the Allow column, select the check box to give Write permissions to IIS_WPG.

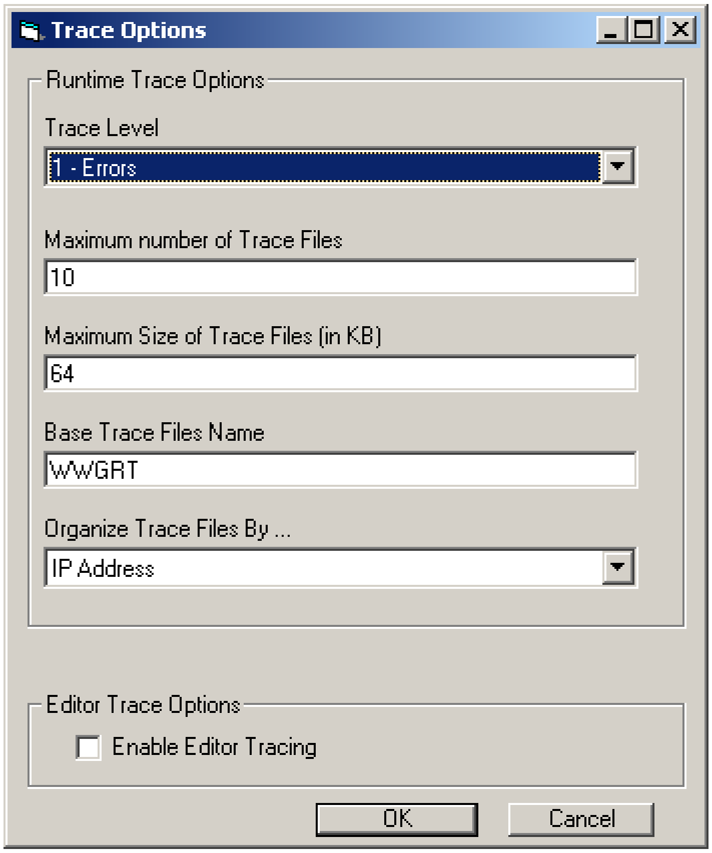
End of procedure

Apache Tomcat (JSP) Deployment

A context is automatically created during Compile time. However, you must restart the Tomcat web server to have it detect the new application for the first time. If you don't restart Tomcat, you will get a message HTTP Status 404 for WWGRun.jsp when trying to start the application.

Tracing the Application

You can set runtime trace options for a script under the menu item Compile > Trace Options. Selecting this command opens the Trace Options dialog box (see Figure 131).



Trace Options

Runtime Trace Options

Trace Level: 1 - Errors

Maximum number of Trace Files: 10

Maximum Size of Trace Files (in KB): 64

Base Trace Files Name: WWGRT

Organize Trace Files By ...: IP Address

Editor Trace Options

☐ Enable Editor Tracing

Figure 131:Trace Options Dialog Box

Table 61describes the controls on the `Trace Options` dialog box.

Table 61: Trace Options

Control	Description / What to Do...
Trace Level	Five trace levels (`0`—`4`) are supported. Valid values are: • `0` (None)—No log events are traced. • `1` (Errors)—Only error conditions are traced. This is the default. • `2` (Warning)—Warning messages are traced. • `3` (Basic Trace)—The normal tracing level, which provides information such as code flow. • `4` (Detailed Trace)—Shows all possible traces, including input/output data. **Note**: Each incremental level also traces the data of the previous level(s).
Maximum number of Trace Files	Number of Trace Files and Size of Trace files. Controls the amount of trace data to be collected that can be modified.
Maximum Size of Trace Files (in KB)	Maximum size of trace files, in kilobytes.
Base Trace Files Name	Enables you to customize trace file names. The default is `WWGRT`. Default file names will have the form `WWGRTxxxxx.log` where `xxxxx` is a sequence number.

Table 61: Trace Options (Continued)

Control	Description / What to Do...
Organize Trace Files By...	Enables you to determine how trace files are to be saved. Each user should have his or her own set of trace files, and the identity of the user can be different depending on the environment. Valid values are IP Address, Cookie ID (Genesys Agent Scripting stores a unique cookie ID on each user's machine), and Host Name.
Enable Editor Tracing	Enables you to trace operations through the Genesys Agent Scripting editor. When this option is selected, the files are stored in the Genesys Agent Scripting Application directory. Two files are created, WWGEditor.log, and WWGEditor.log.bak. • WWGEditor.log contains trace messages indicating what forms have been loaded, and which buttons or menu items the user has selected. This is valid for the current or most recent session. There is no limit on size. • WWGEditor.log.bak contains the data from the previous Editor session.

Additional Logging for Project Books Integrated with the Genesys Agent Interaction Layer

Procedure:
Specify Additional Logging for Project Books

Start of procedure

1. In Configuration Manager, under the Applications folder, go to the application object being used by your Agent Scripting Project Book.

   (This Application object is the same as the one specified for the G_Config_ApplName option, which can be found on the Configuration File tab, under the Compile->Target Environment menu option.)

2. Right-click your Application object and select Properties to open the Properties dialog box.

3. Click the Options tab and double-click the log section.

4. Under the log section, set the file option to debug.

   The default file name is AIL.log.

The default directory for the traces is the directory where the application server is running. For Tomcat, it is the `bin` directory; however, you can specify a path along with the trace file name.

End of procedure

For more information about the tracing options please refer to the *Interaction SDK 7.2 Java Deployment Guide*.

Genesys Agent Scripting Assigner Trace

Trace entries can also be generated by the Genesys Agent Scripting Assigner, which resides within the Desktop client (whether it is `Genesys Agent Desktop` or a custom desktop).

Procedure:
Enable Agent Scripting Assigner Trace

Start of procedure

1. In Configuration Manager, under the `Application` folder, double-click the Application object associated with the desktop application.
2. Click the `Options` tab and then double-click on the `log` section.
3. Add an option called `GAS,` and set the value to `debug`. Also, change the file option to `debug` and specify a file name or path and file name.

   If `GAD` is your desktop, the default directory for the trace files will be `GAD_INSTALL_PATH\logs`.

End of procedure

Example

Procedure:
Example Compile Test

Start of procedure

1. Carry out the steps to set up the small script described in the section “Examples” on page 262 of the chapter “Adding Branching Logic.”
2. Select `Compile` > `Target Environment` to set up a Target Environment for ASP, ASPX, or JSP as described in “Creating a Target Environment” on page 270.
3. Select `Compile` > `Compile,` then select the Target Environment you created in the previous step from the dropdown list.
4. Click the `Compile` button to compile the pages into ASP, ASPX, or JSP code.
5. Start your web server (if you haven’t already done so), and make sure it has a path to the Target Directory identified in the Target Environment.
6. Click the `Simulate` button.
7. In the `Script Launch Page,` select the script you want to run and click the `Simulate` button.

   The first Page of your script should appear in a browser window.
8. Navigate through the script and enter any information requested.
9. Close the browser window when you have finished.

End of procedure

Chapter

13 Advanced Topics

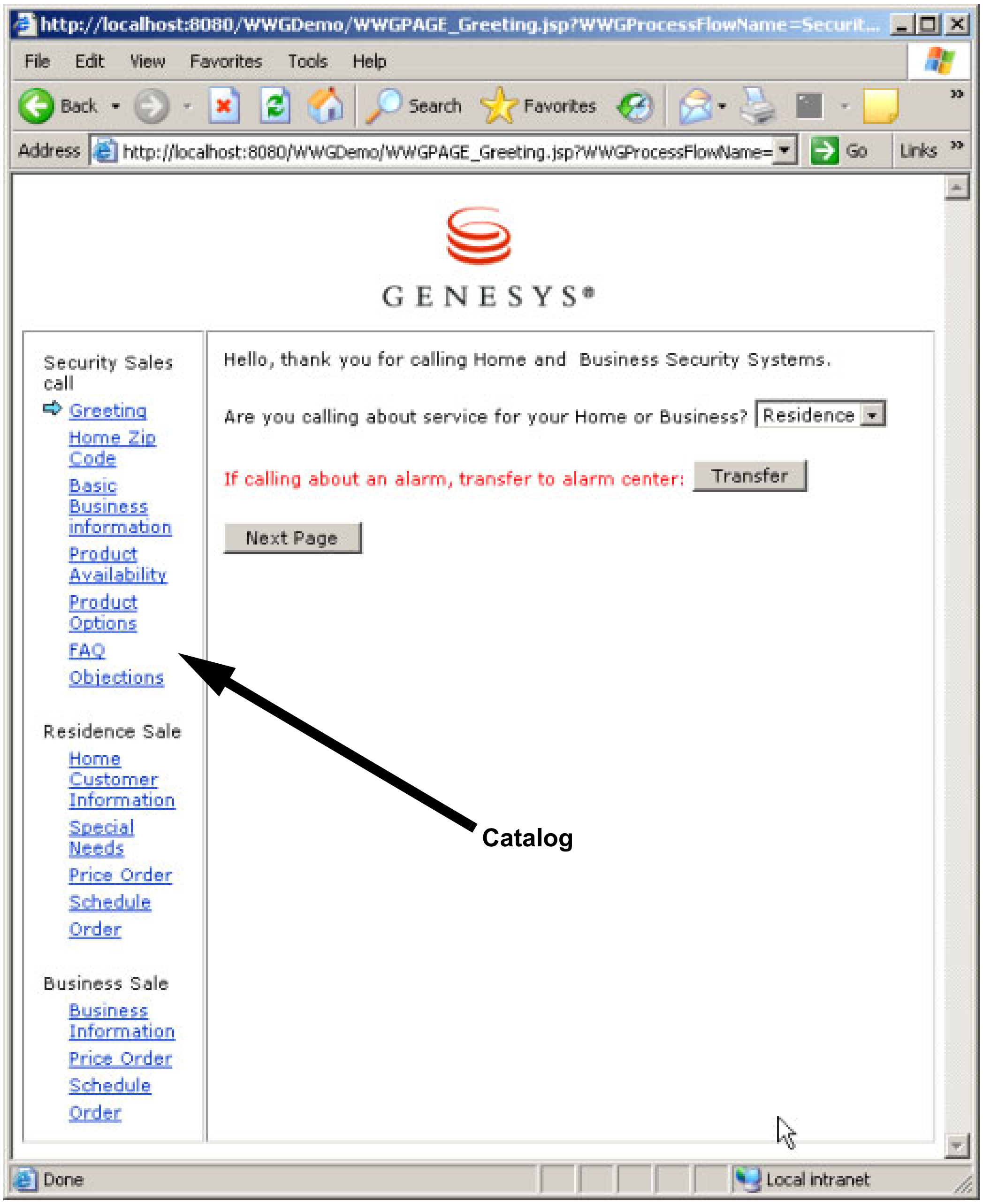
This chapter provides an overview of some additional Genesys Agent Scripting Development Environment features such as creating a Catalog, creating and modifying Page Layouts, importing objects from other Project Books, and managing multiple versions of scripts.

The information in this chapter is divided among the following topics:

Creating a Catalog

A Catalog is a list of links that appears as a navigational tool in the web browser when a script application runs. The Catalog provides a way to move to a Page without following the normal path of an interaction. The Page could be in the same Stream that is currently running, but it does not have to be. In fact, the Page referenced in a Catalog can be from different Streams and Process Flows entirely.

Figure 132 on page 296shows a Catalog as used in the `WWGDemo` Project Book that accompanies the Genesys Agent Scripting Development Environment.



http://localhost:8080/WWGDemo/WWGPAGE_Greeting.jsp?WWGProcessFlowName=Securit...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/WWGDemo/WWGPAGE_Greeting.jsp?WWGProcessFlowName= Go Links
GENESYS

Security Sales call
- Greeting
- Home Zip Code
- Basic Business information
- Product Availability
- Product Options
- FAQ
- Objections

Residence Sale
- Home Customer Information
- Special Needs
- Price Order
- Schedule
- Order

Business Sale
- Business Information
- Price Order
- Schedule
- Order

Hello, thank you for calling Home and Business Security Systems.

Are you calling about service for your Home or Business? Residence

If calling about an alarm, transfer to alarm center: Transfer

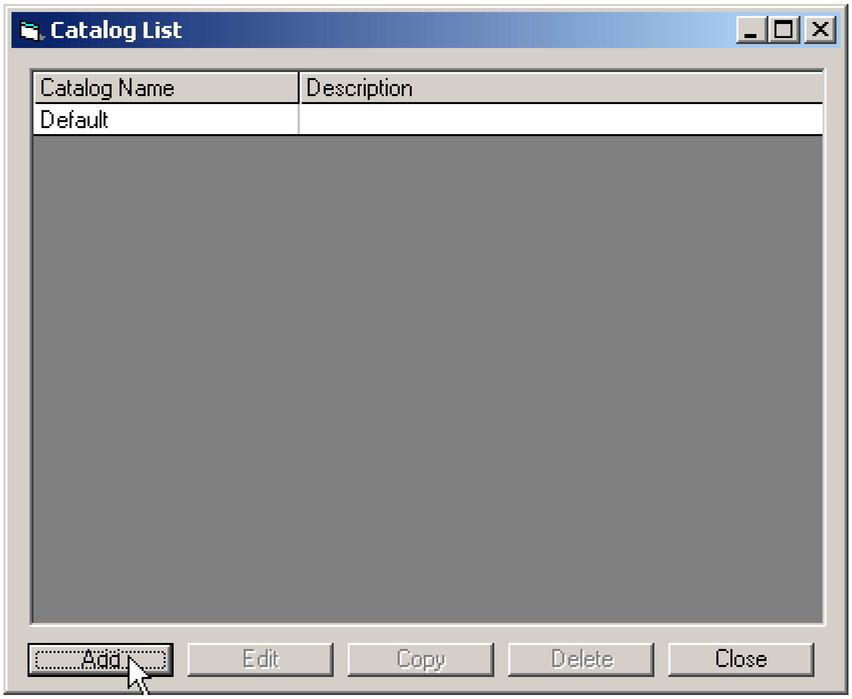
Next Page

Figure 132:Sample Catalog From WWGDemo

Procedure:
Create a new Catalog using the Define menu

Start of procedure

1. Select `Define` > `Catalogs` to open the `Catalog List` dialog box as shown in Figure 133.



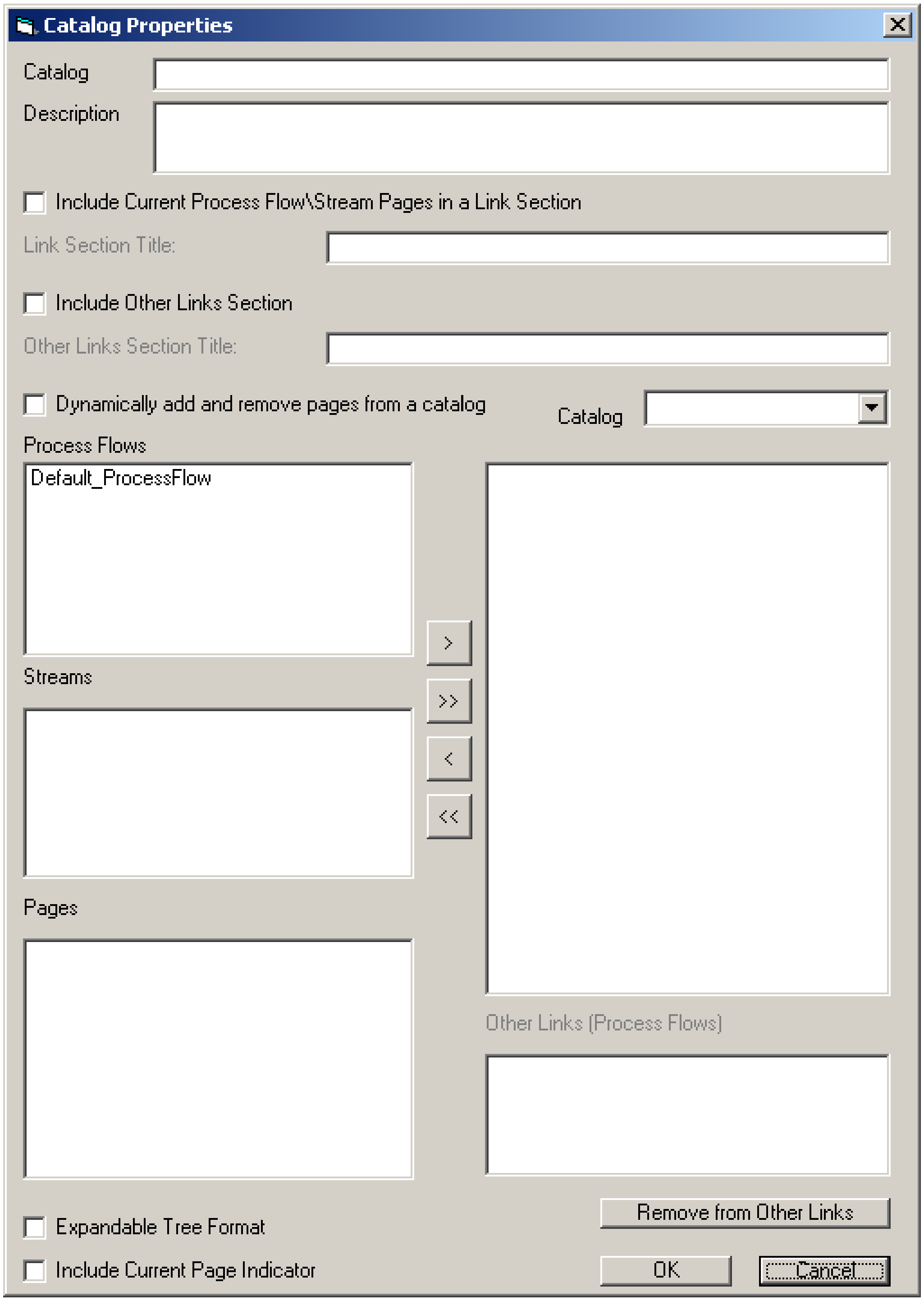
Catalog List

Catalog Name	Description
Default	

Figure 133:Catalog List Dialog Box

2. Click `Add` to initiate the process for creating a new Catalog.

   The `Catalog Properties` dialog box opens as shown in Figure 133.



□ Include Current Process Flow\Stream Pages in a Link Section

□ Include Other Links Section

□ Dynamically add and remove pages from a catalog

Process Flows: Default_ProcessFlow

□ Expandable Tree Format

□ Include Current Page Indicator

Figure 134:Catalog Properties Dialog Box

The `Catalog Properties` dialog box is where you select the Pages that you will add to each Catalog you wish to create.

Drag and drop Process Flows, Streams, or Pages into the Catalog area. You must click and hold until the object is highlighted with a box before dragging the Process Flow, Stream, or Page into the Catalog area.

Table 62 describes the options on the `Catalog Properties` dialog box.

Table 62: Catalog Properties

Control	Description / What to Do...
Catalog	A text field that holds the name value for this Catalog. Type a name for the Catalog you are creating.
Description	The Description is for informational purposes only. This field is not used anywhere else in the system.
Include Current Process Flow\Stream Pages in a Link Section	The Catalog has a link section at the top that contains all Pages in the current script. If you do *not* select this check box, this section of the catalog is suppressed and only selected Pages will appear in the catalog.
Link Section Title	If the `Include Current Process Flow\Stream Pages in a Link Section` check box is selected, this field becomes active and allows you to provide a section title that will appear in the Catalog.
Include Other Links Section	Select this check box to create a section in the Catalog for other links.
Other Links Section Title	If the `Include Other Links Section` check box is selected, this field becomes active and allows you to provide a section title that will appear in the Catalog.
Dynamically add and remove pages from a catalog	Select this checkbox to enable the catalog to duplicate changes that the user makes in the editor (add, delete, and move pages).
Process Flows	You can select which Process Flows shall appear in the Catalog. Once an item is selected, you can drag and drop it into the `Catalog` field. Adding a Process Flow puts all its Streams and Pages into the Catalog.
Streams	You can select which Streams shall appear in the Catalog. Once an item is selected, you can drag and drop it into the `Catalog` field. Adding a Stream puts all its Pages into the Catalog.

Table 62: Catalog Properties (Continued)

Control	Description / What to Do...
Pages	You can select which Pages shall appear in the Catalog. Once an item is selected, you can drag and drop it into the `Catalog` field.
Add All Pages	Click `Add All Pages` to add all the Pages displayed in the Pages list to the Catalog.
Catalog	The `Catalog` field shows the items that will appear in the main section of the Catalog. The `Catalog` dropdown list allows you to display only Pages, only Streams and Pages, or Process Flows, Streams, and Pages in your Catalog, depending on the selection. You will be asked to save changes in the dialog box before changing your view.
Remove From Catalog	Click `Remove From Catalog` to remove items selected in the Catalog list so they are no longer a part of this Catalog.
Other Links (Process Flows)	This shows the links that will appear in the `Other Links` section. Only Process Flows may be added to the `Other Links` section.
Remove From Other Links	Click `Remove From Other Links` to remove items selected in the Other Links (Process Flows) list so they are no longer a part of this Catalog.
Expandable Tree Format	Select this check box so that the links in the Catalog section of the web browser will appear in a *tree* format. Do not select this check box if you want all of the links to appear in a *list* format.
Include Current Page Indicator	Select this check box to cause a pointer to appear next to the link in the Catalog that indicates which Page the script user is currently viewing.
OK	Click `OK` to create the new Catalog or to save changes to the existing Catalog and close the `Catalog Properties` dialog box. Your new catalog is added to the `Catalog List.`

A Catalog must be added to the Page Layout in order for the script user to have access to it. See the next section for information about Page Layouts.

3. Click OK to save your changes.

End of procedure

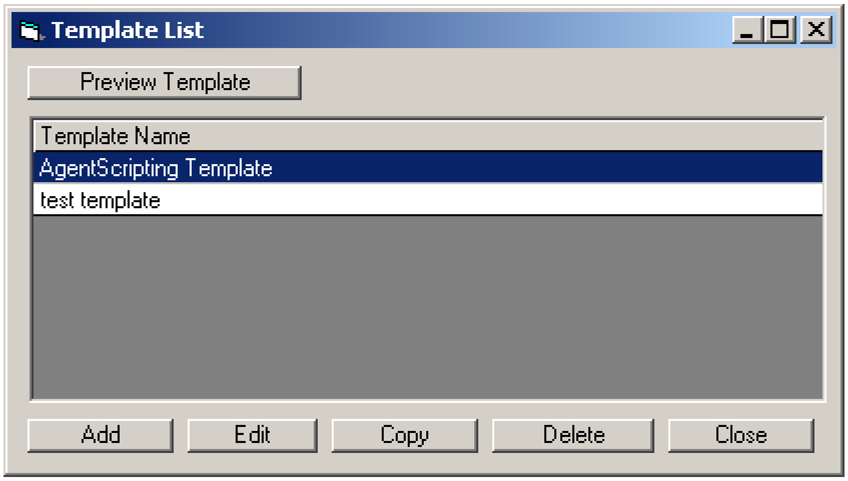
Creating a Template

Templates are useful for setting a common look and feel for all script pages. A Template is an HTML file that is imported into Genesys Agent Scripting. The HTML file can be set up to have the basic formatting that the user wants to have on each page.

Procedure: Create a new Template using the Define menu

Start of procedure

1. Select `Define > Templates` to open the `Template List` dialog box, as shown in Figure 135.



Template Name
AgentScripting Template
test template

Figure 135:Template List Dialog Box

2. Click `Add` to initiate the process for creating a new Template.

   The `Template` dialog box opens, as shown in Figure 136.

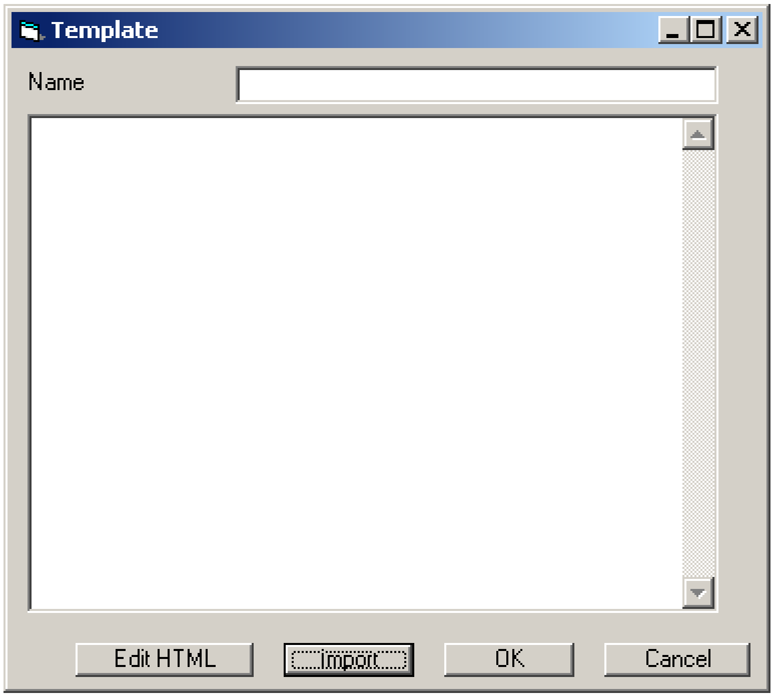


Figure 136:Template Dialog Box

Table 63 describes the properties on the `Template` dialog box.

Table 63: Template Properties

Control	Description / What to Do...
Name	Specify a name in the `Name` text box. This name will appear in the template drop-down list in the `Page Properties` dialog box.
Edit HTML	The `Edit HTML` button allows you to access all HTML that makes up the page. Clicking this button opens the `Edit HTML` dialog box.
View HTML	Click `View HTML` to open the `Edit HTML` dialog box. You can edit the HTML code directly from this dialog box.
Update HTML	Click `Update HTML` to save changes that you have made to the HTML code.
Import	Click `Import` to open the `Select Template` dialog box. From the `Select Template` dialog box, you can browse for a Page Template (.html file) to add to your `Template List`.

3. To apply your new Template to a page, open the `Page Properties` dialog box.
4. From the `Template` drop-down list, select your new Template.

   To preview your new Template, click the `Preview` button. Verify your Template by using the `Template Preview` dialog box. Close the `Template Preview` dialog box by clicking the close button at the top right-hand corner of the dialog box.
5. In the `Page Properties` dialog box, click `OK`.

End of procedure

Creating a Page Layout

A Page Layout describes how Pages will appear to the script user in the production environment. Page Layouts create a unique “look and feel” for scripts and provide an easy way to add an image or style to multiple Pages.

A Page Layout can include different graphical sections, backgrounds, and style sheets. You can create multiple Page Layouts and use them within a single Process Flow or Stream.

By default, Agent Scripting Page Layouts have four sections:

- Header
- Catalog
- Page
- Footer

These sections can be rearranged, resized, or removed from the Page Layout to fit your needs.

Create a new Page Layout using the File menu

- Select **`File`** > **`Page Layouts`** > **`Insert New Page Layout`** to open the **`Page Layout`** dialog box as shown in Figure 137.

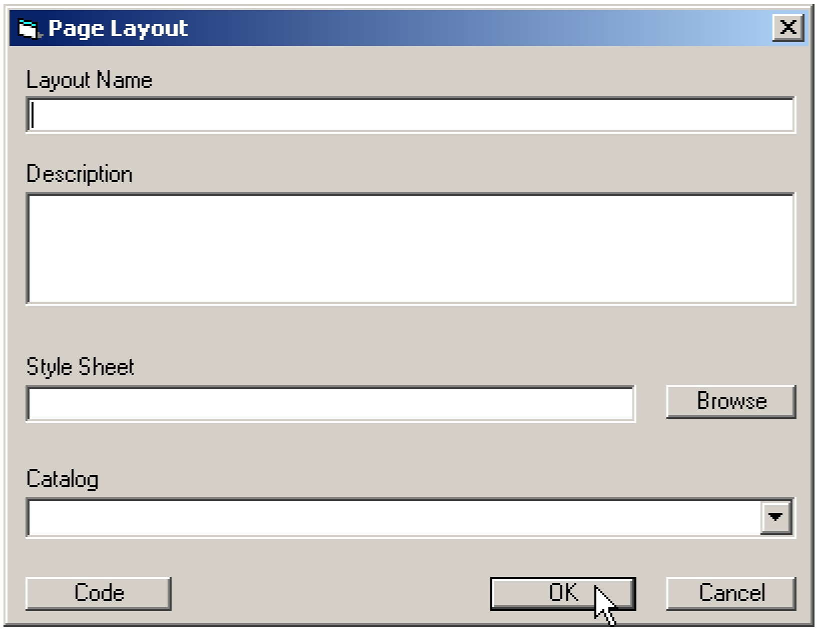


Figure 137:Page Layout Dialog Box

The `Page Layout` dialog box is where you create a new Page Layout and assign a style sheet and Catalog to it.

Table 64 describes the options on the `Page Layout` dialog box.

Table 64: Page Layout

Control	Description / What to Do...
Layout Name	A text field that holds the name value for this Page Layout. This name will appear in a list when you select `File` > `Page Layouts`. Type a name for the Page Layout you are creating.
Description	The Description is for informational purposes only. This field is not used anywhere else in the system.
Style Sheet	You may add a style sheet to the Page Layout. Style sheets are `.css` files defined in the form of rules that tell a web browser how to display specific types of content structures when it encounters these structures in delivering the web page to a user. Type the name of a style sheet file in this field, or find one using `Browse`. New style sheet with release 8.1 is the default, for alignment with Interaction Workspace.

Table 64: Page Layout (Continued)

Control	Description / What to Do...
Browse	Click the Browse button to select a style sheet to add to the Page Layout. Once selected, style sheets are automatically copied to the style sheets directory indicated in the Settings dialog box.
Catalog	You may add a Catalog to the Page Layout. Select a Catalog from the dropdown list.
Code	Click the Code button to create an Action that will run when any Page using this Page Layout is loaded. Actions associated with a Page Layout are run *before* actions associated with a Page. See “Adding Actions to a Page” on page 159 for an example of how a Page Load/Unload action is set up.
OK	Click OK to create the new Page Layout or to save changes to the existing Page Layout and close the Page Layout dialog box.

Using the Page Layout Editor

After creating a Page Layout, you can edit the Page Layout using the File menu and selecting the Page Layout by name.

- Select File > Page Layouts > [name of your layout] to open the Page Layout Editor as shown in Figure 138.

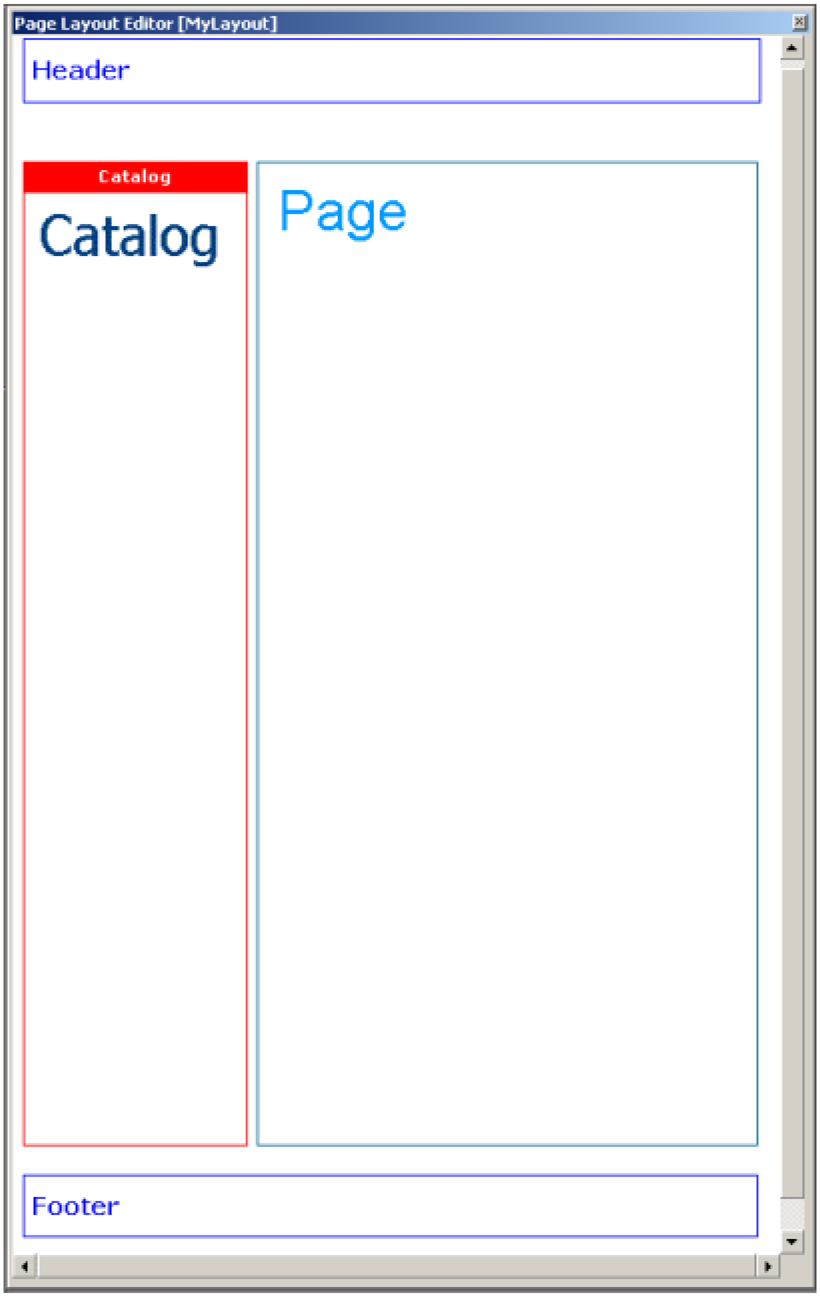


Figure 138:Page Layout Editor

The `Page Layout Editor` replaces the `Page Editor` that you learned about in the "Using the Page Editor" chapter, and the `Page Layout` menu replaces the `Page` menu on the menu bar.

The `Page Layout Editor` works in a similar fashion to the `Page Editor`, except that what you place on the Page Layout will appear on *every Page that uses this Page Layout.*

Adding Fields and Actions to a Page Layout

Place the cursor inside the Page Layout section which will hold the Field or Action. Then, double-click your desired Field in the `Field List`, or your desired Action in the `Action List`, to add it to the Page Layout at that location.

The Field or Action will appear on every Page that uses this Page Layout. For example, if you know you'll want a `Next Page` button on every Page, add it to the Page Layout. If you want to display a particular Field on every Page, add the Field to the Page Layout.

Using the Options on the Page Layout Menu

The `Page Layout` menu allows you to modify elements of your Page Layout to create the unique "look and feel" your application requires. Table 65 shows the options on the `Page Layout` menu and describes how each option is used.

Table 65: Page Layout Menu Options

Option	Description / What it Does...
Delete Page Layout	Select the `Delete Page Layout` menu option to delete the Page Layout currently displayed in the `Page Layout Editor`. You must confirm your request to delete. If any objects currently use this Page Layout, Agent Scripting does not allow the deletion. You can see the objects in a cross reference list, and you must remove references between the objects and this Page Layout before you can delete the Page Layout.
Edit Page Layout Properties	Select the `Edit Page Layout Properties` menu option to open the `Page Layout` dialog box and modify the selected Page Layout. You can then edit any item in this dialog box.
Save As...	Select the `Save As` menu item to save an existing Page Layout under a different name without overwriting the original. The `Page Layout` dialog box appears with a `Save As` button instead of `OK`. Adjust the properties as necessary, then click `Save As` to save a copy of the Page Layout.

Table 65: Page Layout Menu Options (Continued)

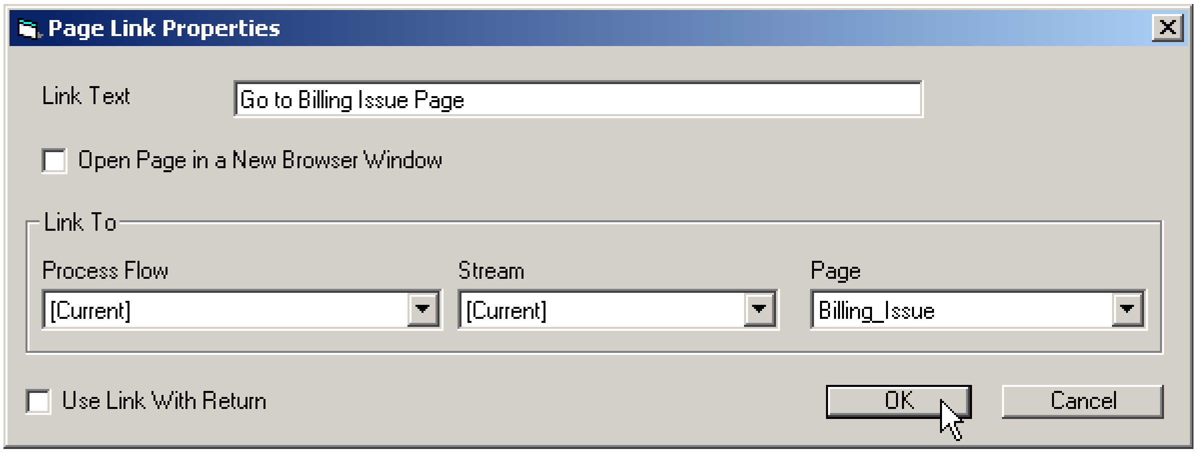
Option	Description / What it Does...
Insert Page Link	Select the `Insert Page Link` menu option to add a link to other Genesys Agent Scripting Pages on the selected Page. This option opens the `Page Link Properties` dialog box. See "Setting Page Link Properties" on page 309 for more information.
Insert Page Area	Select the `Insert Page Area` menu option to add a `Page Area` to the Page Layout. A `Page Area` is where the contents of the created Pages appear in the web browser. You can resize, reposition, or delete `Page Areas` in the Page Layout by selecting the `Page Area` in the `Page Layout Editor`.
Insert Catalog Area	Select the `Insert Catalog Area` menu option to add a `Catalog Area` to the Page Layout. A `Catalog Area` is where a Catalog appears in the web browser with Page navigation information. You can resize, reposition, or delete `Catalog Areas` in the Page Layout by selecting the `Catalog Area` in the `Page Layout Editor`.
Insert Image	Select the `Insert Image` menu option to add an image to a particular area on the Page Layout. This option opens the `Image` dialog box. Once selected, images are automatically copied to the images directory indicated in the `Settings` dialog box. See "Inserting an Image" on page 310 for more information.

Table 65: Page Layout Menu Options (Continued)

Option	Description / What it Does...
Set Background Image	Select the Set Background Image menu option to select a background image to appear in all Pages that use this Page Layout. This option opens the Image dialog box. Once selected, images are automatically copied to the images directory indicated in the Settings dialog box. See "Inserting an Image" on page 310 for more information.
Edit HTML Edit Page Layout View	Select the Edit HTML menu option to access and modify the HTML code that makes up the Page Layout. When you select this option, Genesys Agent Scripting displays *HTML code* in the Page Editor instead of the content of the Page Layout. **Note:** Edit HTML is for advanced, experienced designers only. Use this feature carefully because you may overwrite generated code. The Edit Page Layout View menu option only appears in the Page Layout menu after Edit HTML has been selected. It takes the place of Edit HTML. Select this option to cause Genesys Agent Scripting to display the content of the Page Layout in the Page Layout Editor instead of the HTML code.

Setting Page Link Properties

Figure 139shows the Page Link Properties dialog box.



Link Text: Go to Billing Issue Page

□ Open Page in a New Browser Window

Link To

Process Flow: [Current]

Stream: [Current]

Page: Billing_Issue

□ Use Link With Return

Figure 139:Page Link Properties Dialog Box

Table 66 describes the options on the `Page Link Properties` dialog box.

Table 66: Page Link Properties

Control	Description / What to Do...
Link Text	This field is used to provide the text value that will be displayed as the link text on the selected Page. Type a name for the Page Link you are creating.
Open Page in New Browser Window	Select this check box to indicate that the Page should appear in a new browser window.
Link To:	This areas contains three fields to identify the Process Flow, Stream, and Page to which the link will be associated.
Process Flow	Select a Process Flow from the dropdown list to which this link will be associated.
Stream	Select a Stream from the dropdown list to which this link will be associated.
Page	Select a Page from the dropdown list to which this link will be associated.
Use Link With Return	Select this check box for the Page containing the link to be reloaded upon reaching a return point.
OK	Click `OK` to add the link to the Page and close the `Page Link Properties` dialog box.

Inserting an Image

Figure 140 shows the `Image` dialog box.

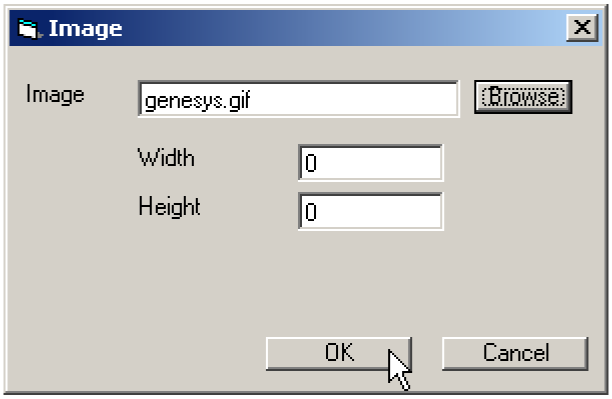


Image: genesys.gif

Width: 0

Height: 0

Figure 140:Image Dialog Box

Table 67 describes the options on the Image dialog box.

Table 67: Image

Control	Description / What to Do...
Image	This field contains the image file name to be added. Type a name for the image file you are adding, or select one using the Browse button.
Browse	Click the Browse button to display a standard Windows dialog box for selecting an image.
Width	Type the width in pixels that the image should occupy. A value of 0 allows the image to occupy the width needed to display it.
Height	Type the height in pixels that the image should occupy. A value of 0 allows the image to occupy the height needed to display it.
OK	Click OK to add the image to the Page Layout, or, if this is a background image, to the Page Layout background.

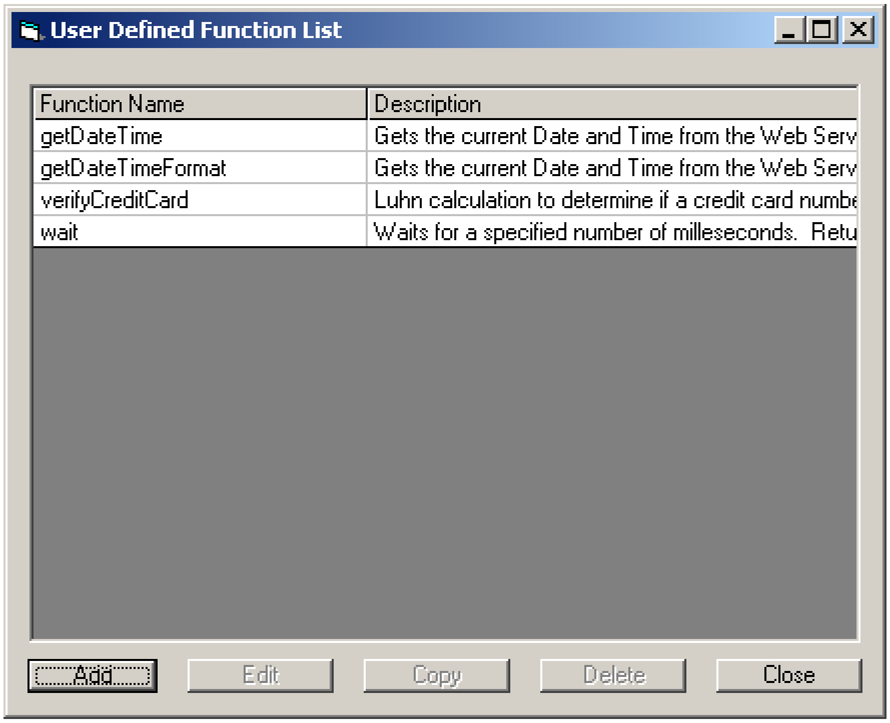
Creating User Defined Functions

Genesys Agent Scripting enables you to add custom functions to the <GAS Install Folder>/ Script xxx Web Folder/ WWGUDFunctions.xxx file (where xxx specifies the target language) for use in your scripts. After you have added your custom code to the WWGUDFunctions.xxx file, you can create a User Defined Function using the File menu.

Procedure: Create a User Defined Function using the File menu

Start of procedure

1. Select `File > user Defined Functions` to open the `User Defined Function List` dialog box as shown in Figure 141.

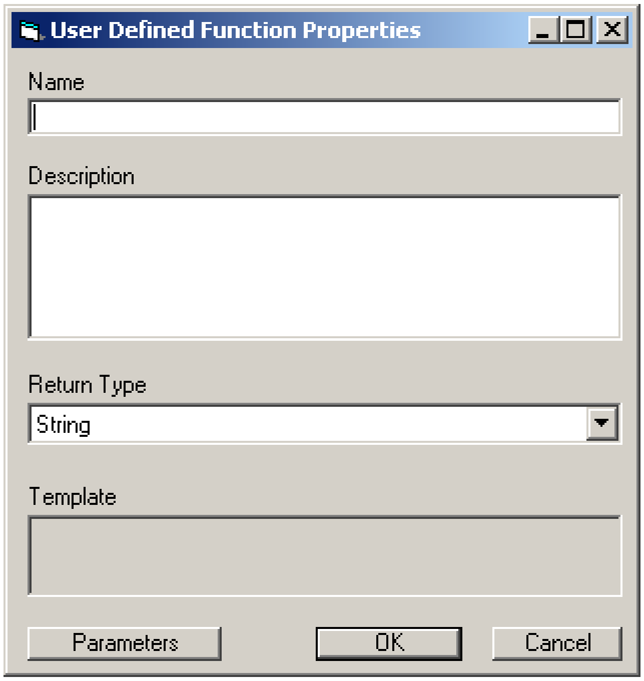


Function Name	Description
getDateTime	Gets the current Date and Time from the Web Serv
getDateTimeFormat	Gets the current Date and Time from the Web Serv
verifyCreditCard	Luhn calculation to determine if a credit card numbe
wait	Waits for a specified number of milleseconds. Retu

Figure 141:User Defined Function List Dialog Box

2. Click `Add` to initiate the process for creating a new User Defined Function.

   The `User Defined Function Properties` dialog box opens, as shown in Figure 142.



User Defined Function Properties

Name

Description

Return Type
String

Template

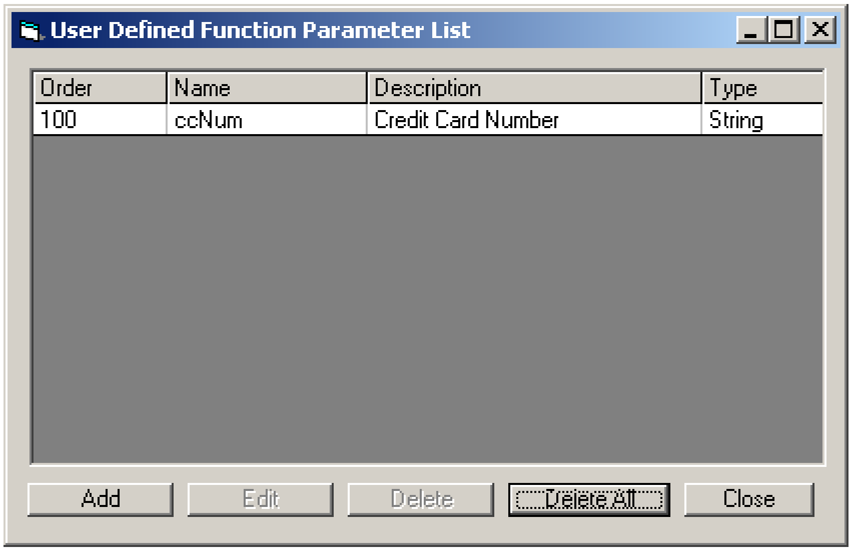
Figure 142:User Defined Function Properties Dialog Box

Table 68 describes the options on the `User Defined Function Properties` dialog box.

Table 68: User Defined Function Properties

Control	Description / What to Do...
Name	Specifies a name for the User Defined Function.
Description	Specifies a description of the function.
Return Type	From the drop-down list, select the format of the data returned by the function. Valid values are `String`, `Number`, `Boolean`, and `DateTime`.
Template	The template is generated based on the name and parameters assigned to the function. This is a display-only field.
Parameters	Information passed into the function. Click the `Parameters` button to open the `User Defined Function Parameter List` dialog box.

3. To define parameters to be passed into your function, click the `Parameters` button to open the `User Defined Function Parameter List` dialog box (see Figure 143).

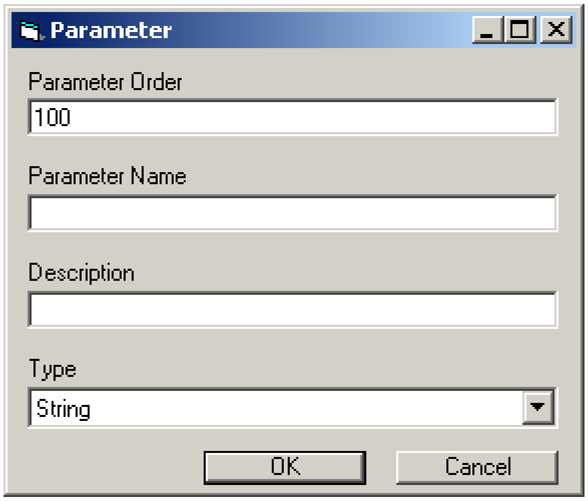


Order	Name	Description	Type
100	ccNum	Credit Card Number	String

Figure 143:User Defined Function Parameter List Dialog Box

4. Click `Add` to initiate the process for creating parameters for your User Defined Function.

   The `Parameter` dialog box opens, as shown in Figure 144.



Parameter Order: 100

Parameter Name:

Description:

Type: String

Figure 144:Parameter Dialog Box

Table 69 describes the options on the `Parameter` dialog box.

Table 69: User Defined Function Parameters

Control	Description / What to Do...
Parameter Order	Specifies the order in which parameters will be passed to the function.
Parameter Name	Name of the parameter being passed to the function.
Description	Description of the parameter.
Type	Specifies the format of the data being passed to the function. Use the drop-down list to select the appropriate `Type`. Valid values are: • `String`. • `Number`. • `Boolean`. • `DateTime`.

5. Add the parameters required for your function.

End of procedure

Using a User Defined Function

Note: The following example of how to use a User Defined Function is based on calling a function named `verifyCreditCard` that verifies that the credit card number is valid, and then does something based on a successful verification.

To use your User Defined Function in a script, you must create an Action (refer to “Creating an Action” on page 110 for details) and write custom code (refer to “Code for Action” on page 130 for details) for the Action. When the Action is used in your script, it will execute the custom code defined in the Code for Action dialog box, and call your User Defined Function.

Procedure: Create an Action that will execute a User Defined Function

Start of procedure

1. On the Action List dialog box, click Add to initiate the process for creating a new Action.
2. On the Action Wizard dialog box, click Skip Wizard to open the Action Properties dialog box.
3. On the General tab of the Action Properties dialog box:
   - In the Button Text field enter a name for the Button that will be used to call your User Defined Function.
   - In the Name field, enter a name for the Action that you are creating.
   - From the Action Type drop-down list, select Function.
4. Click the Code button to open the Code for Action dialog box. The section "Code for Action" on page 130 contains detailed information about this dialog box.
5. Use the Insert After and Insert Before buttons to arrange the order for each line of your custom code.

   The following is a step-by-step procedure for adding custom code in the Code for Action dialog box. These steps do not contain a full working code sample, but rather illustrate how to insert custom code by using the GUI. The custom code in this example calls the function verifyCreditCard().

   verifyCreditCard() accepts a credit card number (as type String) as input. If the credit card number is valid, the function will return a "1" and perform additional processing, otherwise, the function will return a "0" indicating that the credit card number is invalid.

   It should be noted that defining and initializing variables are required only if the User Defined Function has parameters. In this example, the verifyCreditCard() function takes a String as an input parameter. This input parameter is passed to the function in a String variable called ccNum.

   Defining a variable:

   a. Click Insert After to open the first line in the Code for Action main window.

   b. In the Command window, select DIM to create a variable that will contain the credit card number that will be passed to the User Defined Function.

   c. Under the Free Form drop-down list, type ccNum.

   d. In the Operator window, select AS.

e. Under the `Type` drop-down list, select `String`.

Initializing the variable:

f. Click `Insert After` to open the second line in the `Code for Action` main window.

g. In the `Command` window, select `SET`.

h. From the first drop-down list, select `Free Form`, and in the `Free Form` window type `ccNum`.

i. In the `Operator` window, select `=`.

j. From the second drop-down list, select `Free Form`, and in the `Free Form` window, specify a series of zeros (`000000000`) to initialize the `ccNum` variable.

Calling the User Defined Function:

k. Click `Insert After` to open another line in the `Code for Action` main window.

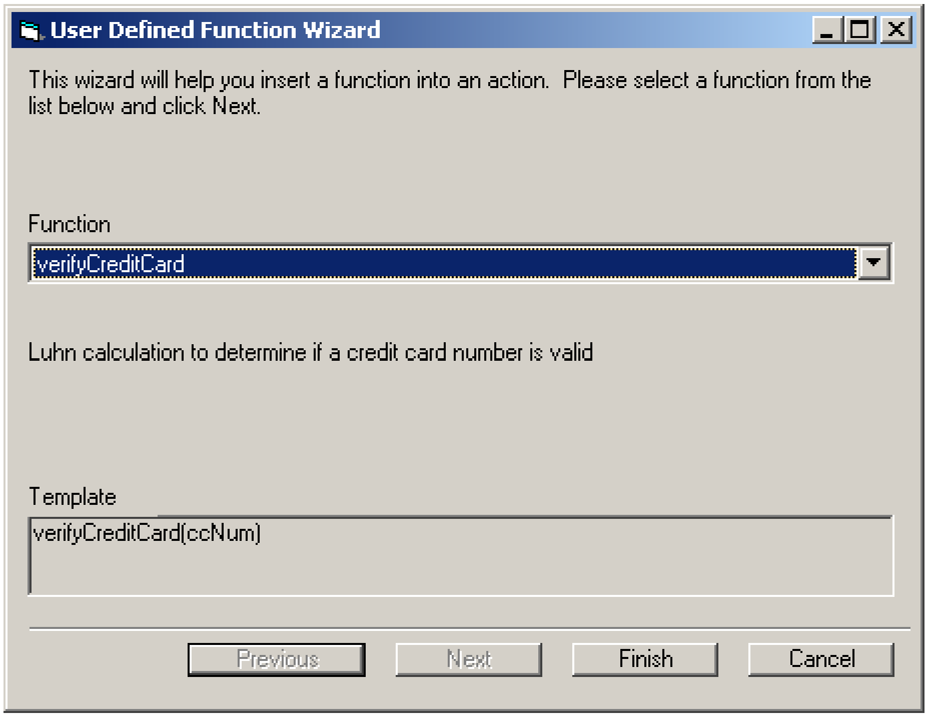
l. In the `Command` window, select `IF`.

m. From the first drop-down list, select `Free Form`, and in the `Free Form` window type “`1`" in quotation marks.

n. In the `Operator` window, select `=`.

o. From the second drop-down list, select `Free Form`. The `Insert Function` button appears.

p. Click `Insert Function` to open the `User Defined Function Wizard` (see Figure 145).



User Defined Function Wizard

This wizard will help you insert a function into an action. Please select a function from the list below and click Next.

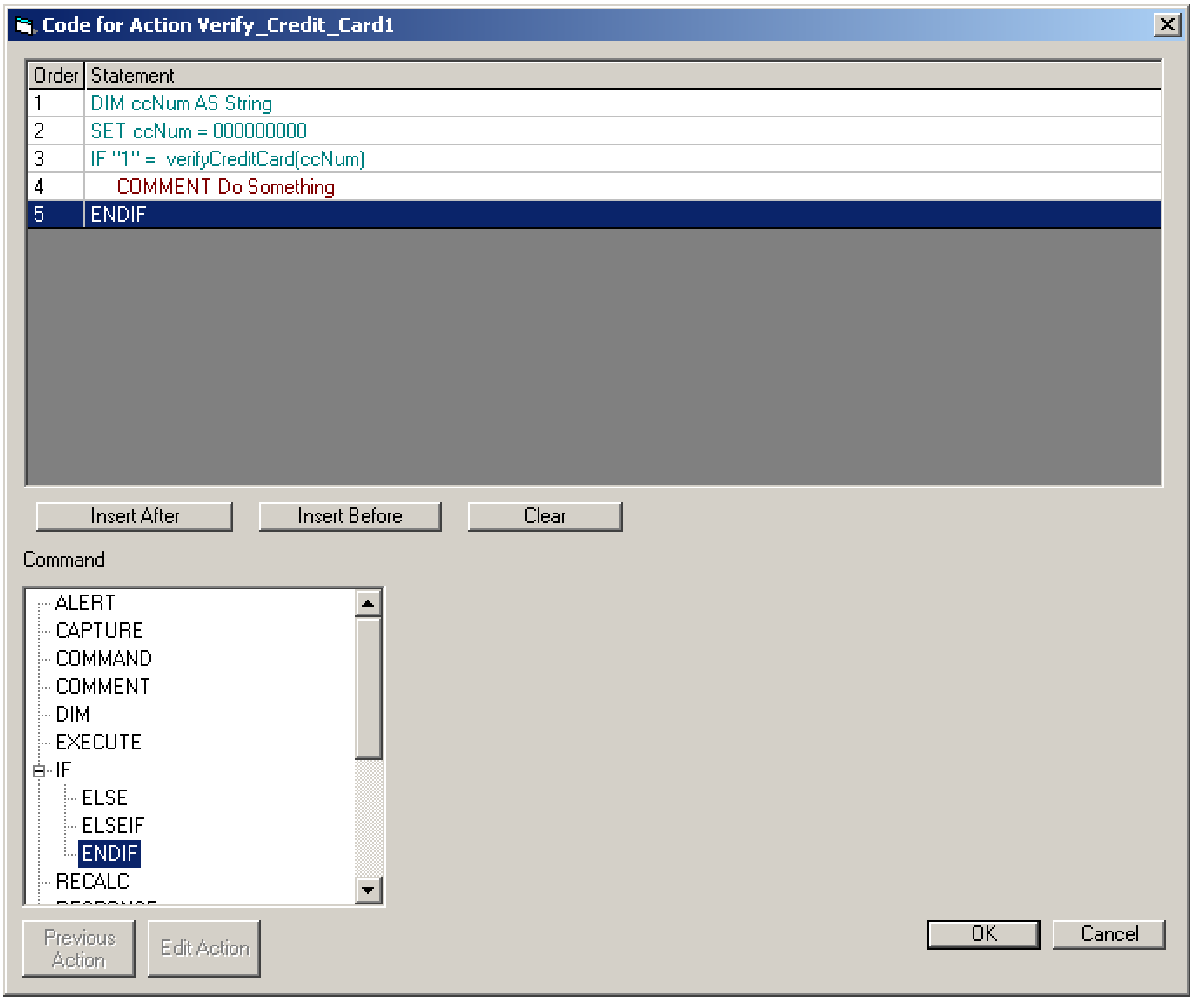
Function: verifyCreditCard

Luhn calculation to determine if a credit card number is valid

Template: verifyCreditCard(ccNum)

Figure 145:User Defined Function Wizard

6. From the Function drop-down list, select your User Defined Function.
7. Click Finish to add your User Defined Function and return to the Code for Action dialog box.
8. Click Insert After to open another line in the Code for Action main window.
9. In the Command window, select Comment. This line is illustrating, with the use of the Comment command, that additional logic could be written within the logical IF condition.
10. From the drop-down list, select Free Form, and in the Free Form window, type Do Something.
11. To close the IF condition, in the Command window, expand IF and select EndIF.The resulting code is shown in the Code for Action dialog box (see Figure 146).



Order	Statement
1	DIM ccNum AS String
2	SET ccNum = 000000000
3	IF "1" = verifyCreditCard(ccNum)
4	COMMENT Do Something
5	ENDIF

Figure 146:Code for Action Dialog Box—Custom Code Added

12. Click OK to return to the Action Properties dialog box.

13. On the `Action Properties` dialog box, click `OK` to add your User Defined Function to the `Action List`.

End of procedure

Creating Custom Field Types

Custom Field Types enable you to define field types that are specific to your needs. For each Custom Field Type, you can determine which of the target languages (ASP, ASPX, or JSP) are supported, as well as how the field will be displayed, and how it will process form data for the supported languages.

Procedure: Create a Custom Field Type using the Define menu

Start of procedure

1. Select `Define > Custom Field Types` to open the `Custom-Field-Type List` dialog box, as shown in Figure 147.

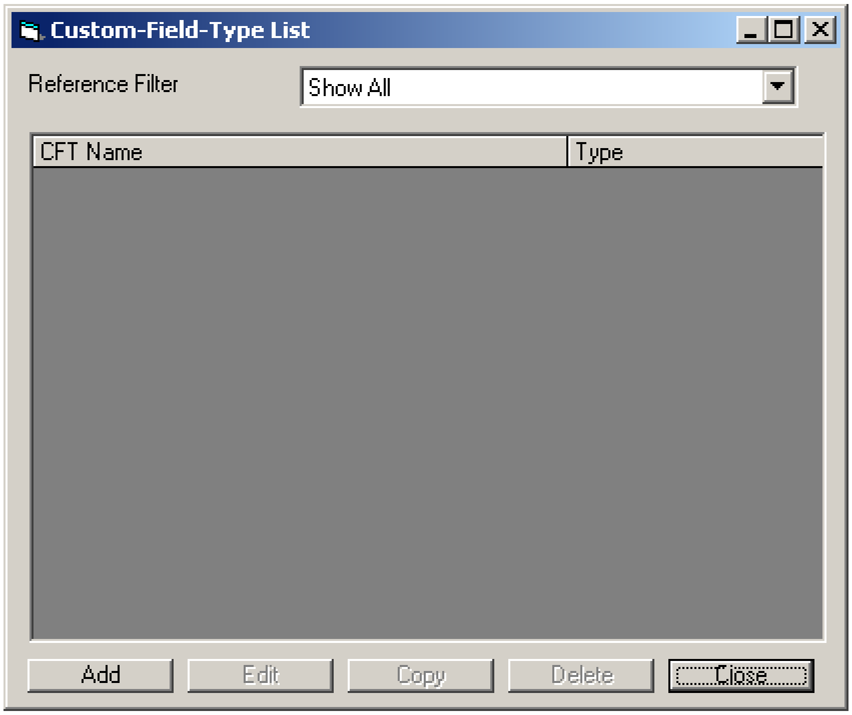
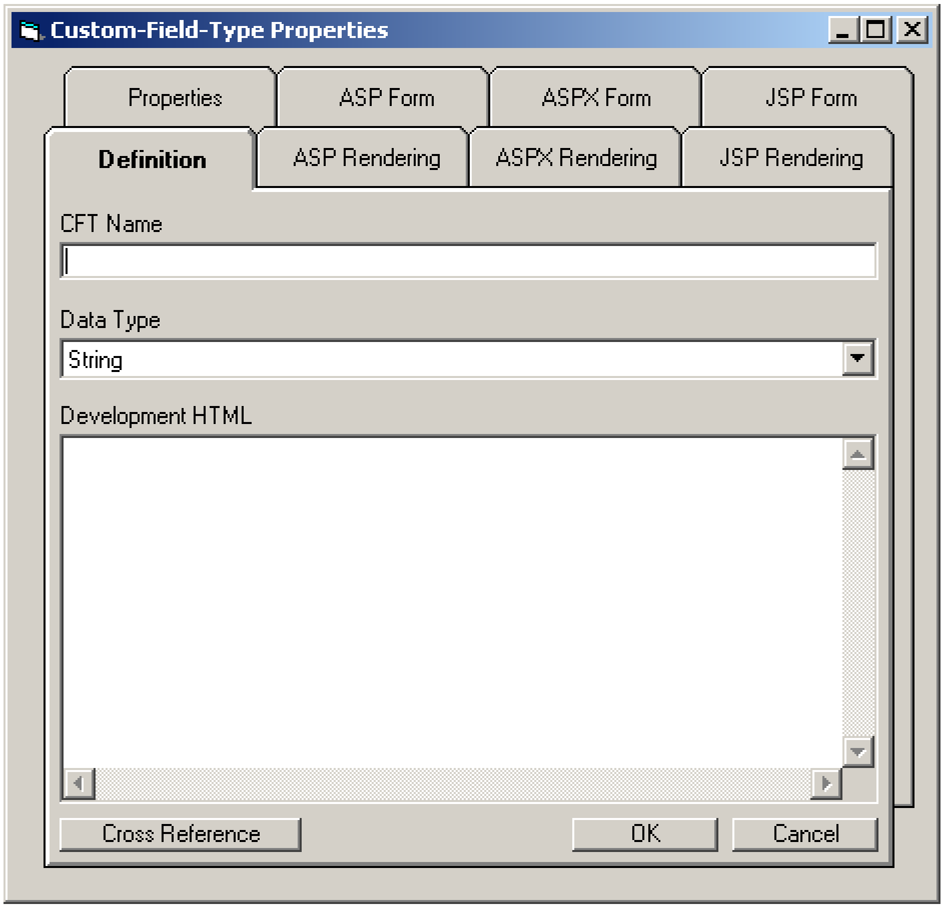


Figure 147:Custom-Field-Type List Dialog Box

2. Click `Add` to initiate the process for creating a new Custom Field Type.

3. The Custom-Field-Type Properties dialog box opens as shown in Figure 148.



Custom-Field-Type Properties

Definition tab

CFT Name:

Data Type: String

Development HTML:

Figure 148:Custom-Field-Type Properties Dialog Box—Definition Tab

You use the Custom-Field-Type Properties dialog box to define the characteristics that are unique to each Custom Field Type that you wish to create. This dialog box contains eight tabs, each of which is described in turn in the following sections.

End of procedure

Definition Tab

The Definition tab is the default view of the Custom-Field-Type Properties dialog box. This tab enables you to define the Custom Field Type with respect to its Genesys Agent Scripting behaviors.

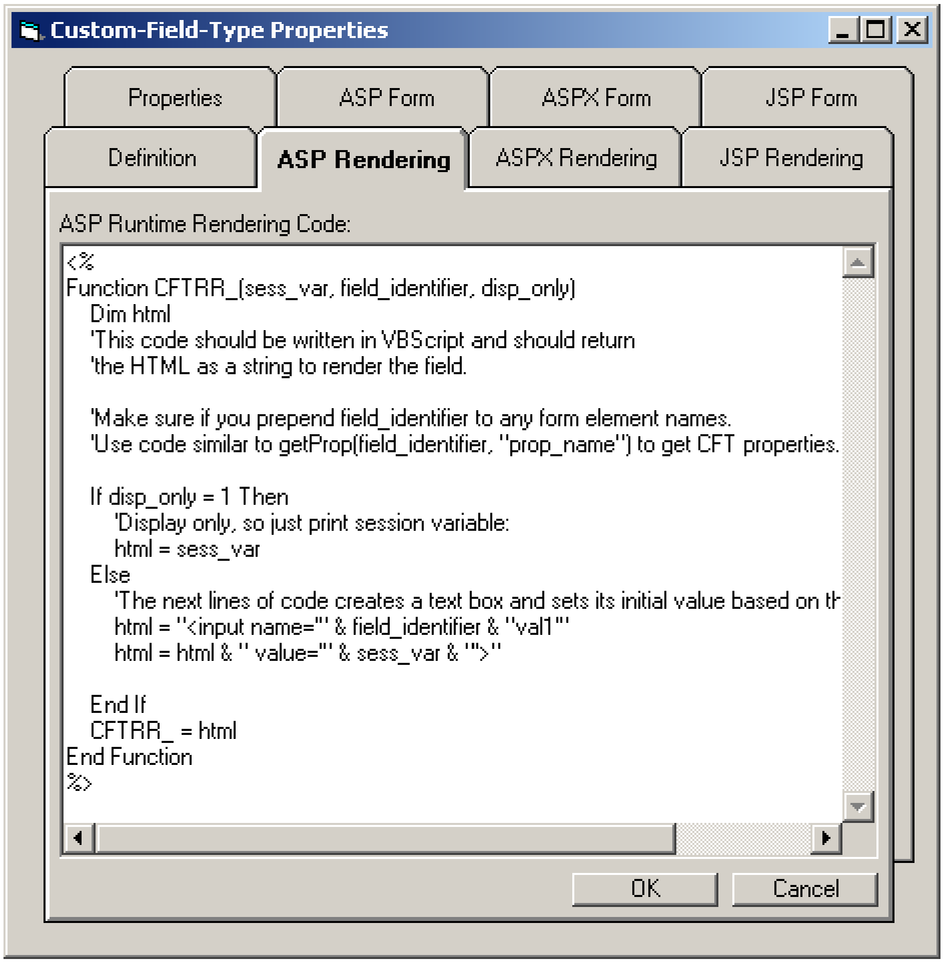
Table 70 describes the options on the Definition tab of the Custom-Field-Type Properties dialog box.

Table 70: User Defined Function Properties

Control	Description / What to Do...
CFT Name	Identifies the `Custom Field Type` within Genesys Agent Scripting. The `CFT Name` must be unique, and it cannot contain any spaces.
Data Type	Describes how the data is stored internally within the session. Select one of the following Data Types from the drop-down list: • `String` • `Numeric` • `Date and Time` • `Boolean`
Development HTML	Enables you to define how the custom field type looks within the editor. An example of Development HTML code is: `<input size=%size% value="test">` The Development HTML generally works in the same manner as conventional HTML. The only difference is that properties defined on the `Properties` tab can be assigned to attributes in a tag. To do this, the property must be enclosed by the % (percent sign), as shown by the size attribute and its assigned property in the preceding example of an `<input>` tag. You can enter only one top-level node in the `Development HTML` field. The end tag must either correspond to the first tag, as in `<table>` and `</table>` tags, or there must be only one tag, with no end tag, as in the `<input>` tag example.
Cross Reference	Click to open the `Object Cross Reference` dialog box. The `Object Cross Reference` dialog box lists all fields that reference the `Custom Field Type`.

ASP Rendering Tab

Figure 149 shows the `ASP Rendering` tab of the `Custom-Field-Type Properties` dialog box.



```
<%
Function CFTRR_(sess_var, field_identifier, disp_only)
   Dim html
   'This code should be written in VBScript and should return
   'the HTML as a string to render the field.

   'Make sure if you prepend field_identifier to any form element names.
   'Use code similar to getProp(field_identifier, "prop_name") to get CFT properties.

   If disp_only = 1 Then
      'Display only, so just print session variable:
      html = sess_var
   Else
      'The next lines of code creates a text box and sets its initial value based on th
      html = "<input name=""" & field_identifier & "val1"""
      html = html & " value=""" & sess_var & """>"

   End If
   CFTRR_ = html
End Function
%>
```

Figure 149:Custom-Field-Type Properties Dialog Box—ASP Rendering Tab

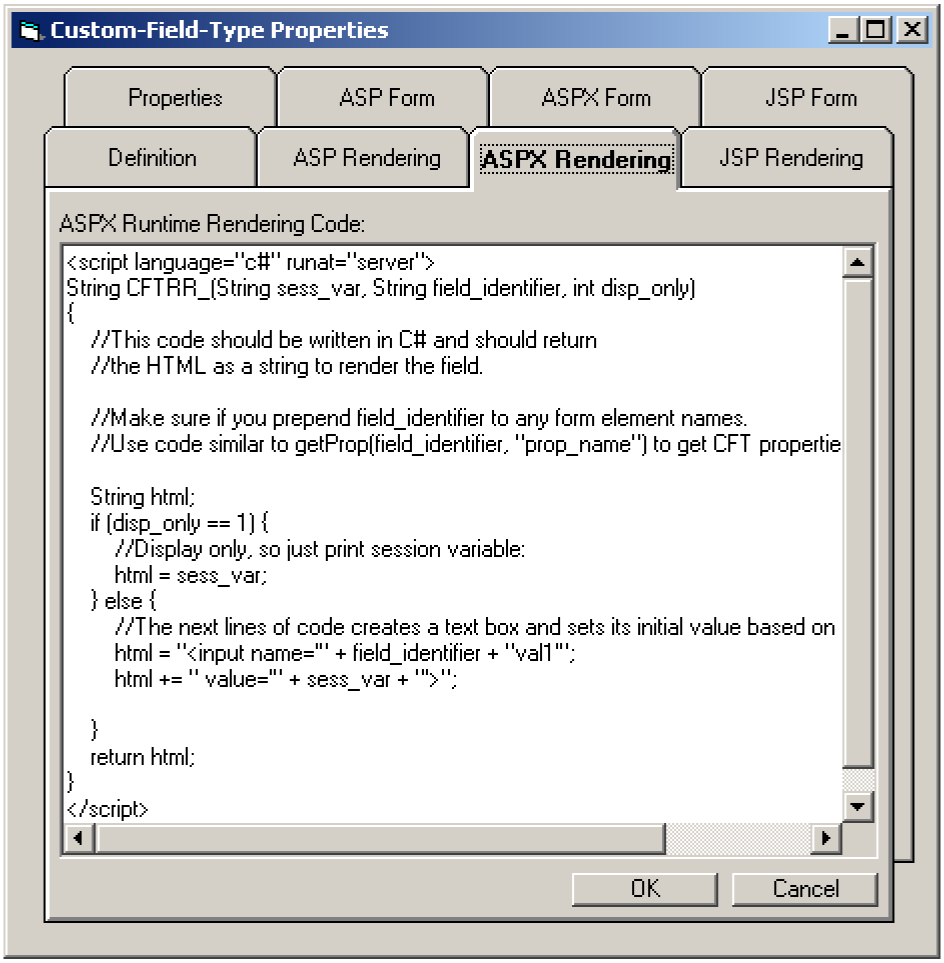
The `ASP Rendering` tab determines how the field will look when it is compiled into ASP code when viewed in a web browser. ASP Rendering Code is Visual Basic code that takes in as arguments the session variable for the instance of the Custom Field Type, the field identifier, and the display-only indicator. The field identifier is a unique value determined by Genesys Agent Scripting to identify a field, and it is prefixed to any form variables.

The example code shown on the `ASP Rendering` tab is a function that creates an `<input>` tag containing the name, size, and value attributes.

Properties defined on the `Properties` tab can be assigned to attributes in a tag in the ASP Rendering Code. To do this, the property is referenced by the `getProp()` function call, as seen by the size attribute and its assigned property in the example code shown in Figure 149.

ASPX Rendering Tab

Figure 150shows the `ASPX Rendering` tab of the `Custom-Field-Type Properties` dialog box.



ASPX Runtime Rendering Code:

```
<script language="c#" runat="server">
String CFTRR_(String sess_var, String field_identifier, int disp_only)
{
   //This code should be written in C# and should return
   //the HTML as a string to render the field.

   //Make sure if you prepend field_identifier to any form element names.
   //Use code similar to getProp(field_identifier, "prop_name") to get CFT propertie

   String html;
   if (disp_only == 1) {
      //Display only, so just print session variable:
      html = sess_var;
   } else {
      //The next lines of code creates a text box and sets its initial value based on
      html = "<input name='" + field_identifier + "val1'";
      html += " value='" + sess_var + "'>";

   }
   return html;
}
</script>
```

Figure 150:Custom-Field-Type Properties Dialog Box—ASPX Rendering Tab

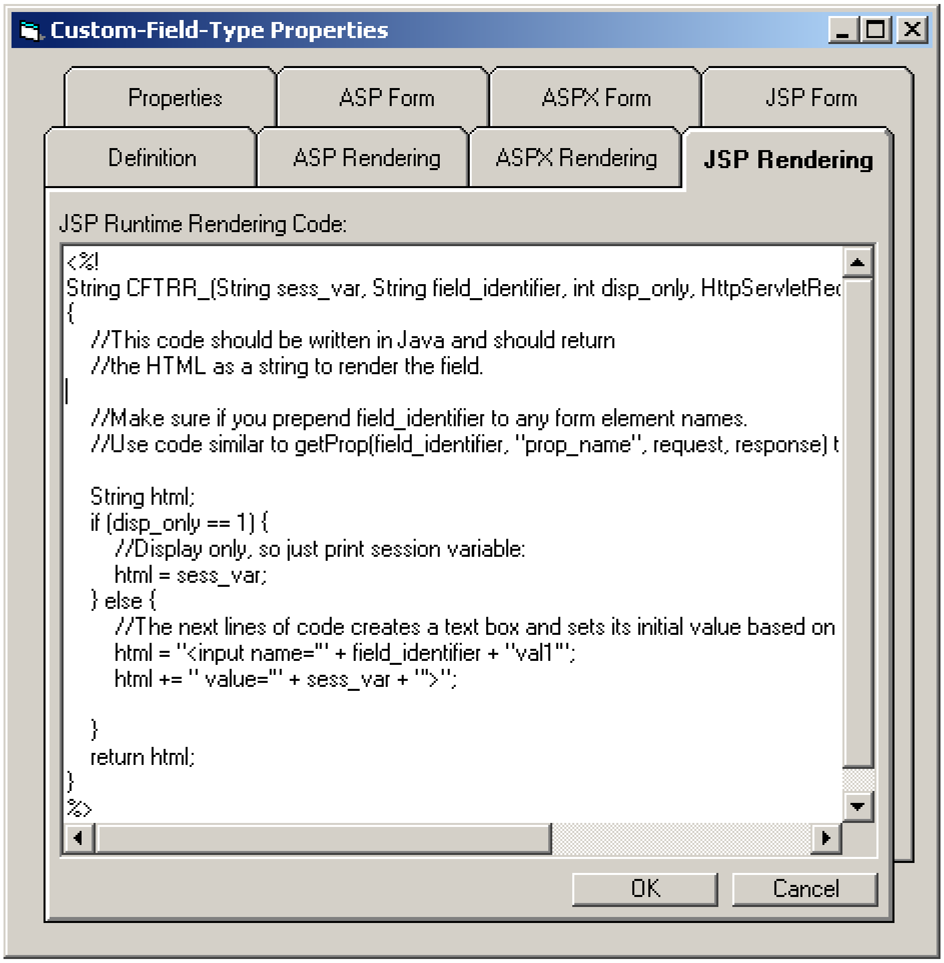
The `ASPX Rendering` tab determines how the field will look when compiled into ASPX code when viewed in a web browser. The ASPX Rendering Code is C# code that takes in as arguments the session variable for the instance of the Custom Field Type, the field identifier, and the display-only indicator.

The example code shown on the `ASPX Rendering` tab is a function that creates an `<input>` tag containing the name, size, and value attributes.

Properties defined on the `Properties` tab can be assigned to attributes in a tag in the ASPX Rendering Code. To do this, the property is referenced by the `getProp()` function call, as seen by the size attribute and its assigned property above in the example code shown in Figure 150.

JSP Rendering Tab

Figure 151shows the `JSP Rendering` tab of the `Custom-Field-Type Properties` dialog box.



```
<%!
String CFTRR_(String sess_var, String field_identifier, int disp_only, HttpServletRec
{
   //This code should be written in Java and should return
   //the HTML as a string to render the field.

   //Make sure if you prepend field_identifier to any form element names.
   //Use code similar to getProp(field_identifier, "prop_name", request, response) t

   String html;
   if (disp_only == 1) {
      //Display only, so just print session variable:
      html = sess_var;
   } else {
      //The next lines of code creates a text box and sets its initial value based on
      html = "<input name='" + field_identifier + "val1'";
      html += " value='" + sess_var + "'>";

   }
   return html;
}
%>
```

Figure 151:Custom-Field-Type Properties Dialog Box—JSP Rendering Tab

The `JSP Rendering` tab determines how the field will look when compiled into JSP code when viewed in a web browser. The JSP Rendering Code is Java code that takes in as arguments the session variable for the instance of the Custom Field Type, the field identifier, the display-only indicator, the HttpServletRequest, and the HttpServletResponse.

The example code shown on the `JSP Rendering` tab is a function that creates an `<input>` tag containing the name, size, and value attributes.

Properties defined on the `Properties` tab can be assigned to attributes in a tag in the JSP Rendering Code. To do this, the property is referenced by the `getProp()` function call, as seen by the size attribute and its assigned property in the example code shown in Figure 151.

Properties Tab

Figure 152 shows the Properties tab of the Custom-Field-Type Properties dialog box.

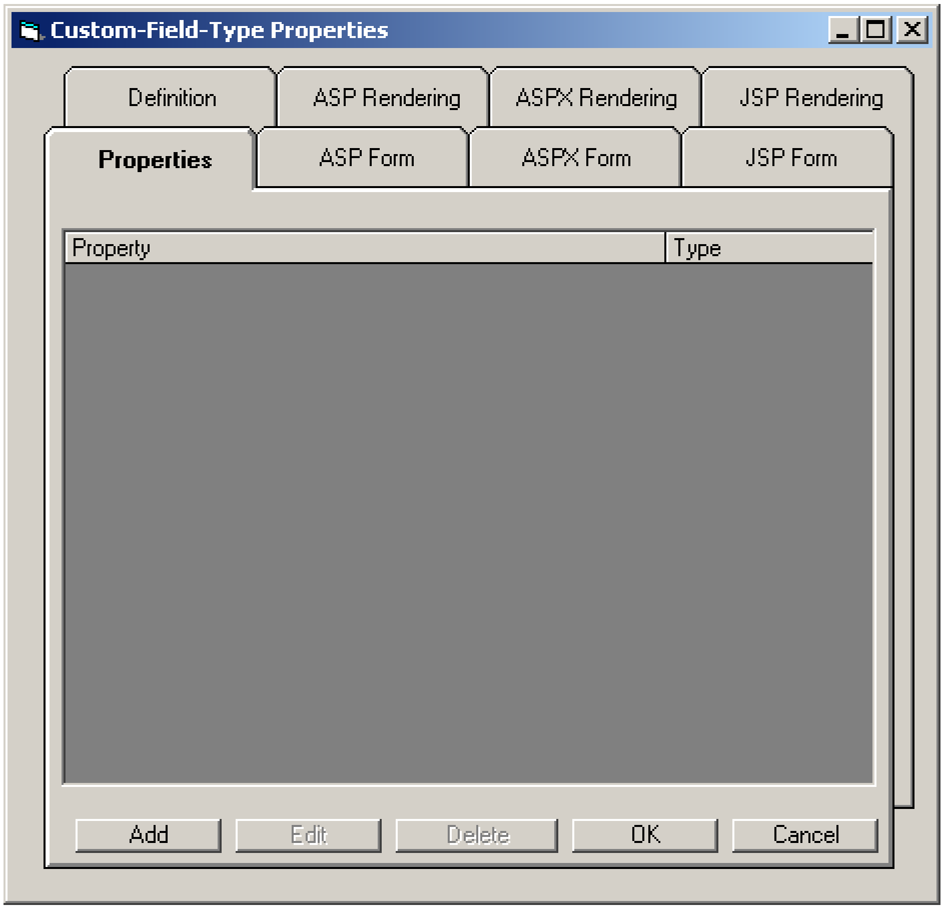
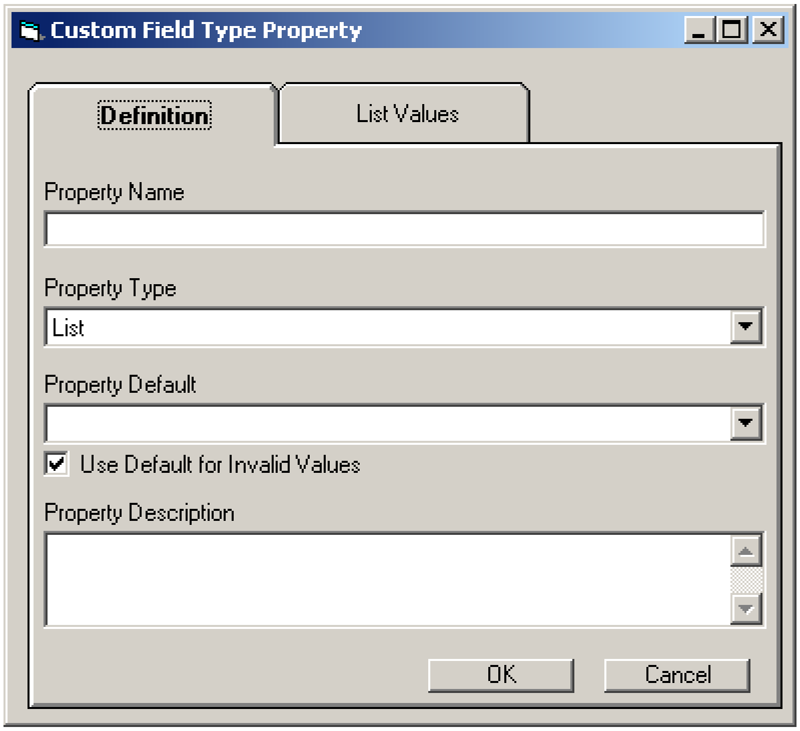


Figure 152:Custom-Field-Type Properties Dialog Box—Properties Tab

Properties describe the behavior of the Custom Field Type, both within Genesys Agent Scripting and in the defined target languages.

When referenced in the Development HTML field on the Definition tab, properties are enclosed by the percent sign (%). The getProp() function call is used to reference properties on the ASP, ASPX, and JSP Rendering and Form tabs.

Click Add to open the Custom Field Type Property dialog box shown in Figure 153.



Property Name

Property Type: List

Property Default

☑ Use Default for Invalid Values

Property Description

Figure 153:Custom Field Type Property Dialog Box

Table 71 describes the options on the `Definition` tab of the `Custom Field Type Property` dialog box.

Table 71: Custom Field Type Property Options

Control	Description / What to Do...
Definition Tab	Contains the fields that are used to define the property.
Property Name	Name of the property. The `Property Name` must follow these rules: • Must be unique within the Custom Field Type. • Cannot contain spaces. • Cannot contain special characters.
Property Type	Describes the property's value. Valid values in the drop-down list are: • `String.` • `Numeric.` • `Boolean.` • `List.`

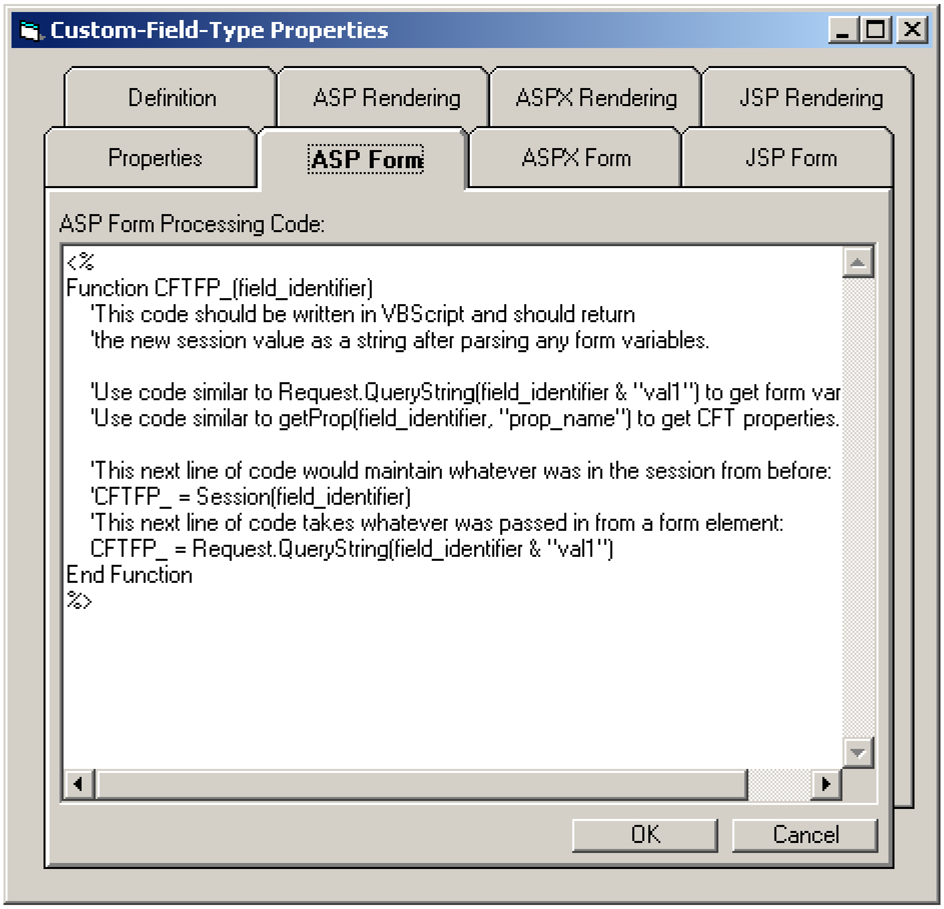
Table 71: Custom Field Type Property Options (Continued)

Control	Description / What to Do...
Property Default	Specifies the default value for the property. This field is a text box for `String` and `Numeric` property types, and a drop-down list for `Boolean` and `List` types.
Use Default for Invalid Values	If this check box is selected, the default value entered in the `Property Default` field becomes the property's value if an invalid one is entered on the `Custom Type` tab of the Field Properties.
Property Description	Enter a description of the property. This is for reference only. The description is not found anywhere else in the system.
List Values Tab	Contains a field for specifying list values.
List Values	Specify a set of list items to select in the `Property Default` field, and in the `Custom Type` tab of the Field Properties. To create a list of values, enter the values in this field and press `Return` between each item in the list.

ASP Form Tab

The ASP Form Process code takes what is submitted by the form, and returns the session variable with its new value.

The ASP Form Code is Visual Basic code that takes in the field identifier as an argument, and returns a new value for the session variable. Figure 154 shows an example of code that might be entered on the `ASP Form` tab.



ASP Form Processing Code:

```
<%
Function CFTFP_(field_identifier)
   'This code should be written in VBScript and should return
   'the new session value as a string after parsing any form variables.

   'Use code similar to Request.QueryString(field_identifier & "val1") to get form var
   'Use code similar to getProp(field_identifier, "prop_name") to get CFT properties.

   'This next line of code would maintain whatever was in the session from before:
   'CFTFP_ = Session(field_identifier)
   'This next line of code takes whatever was passed in from a form element:
   CFTFP_ = Request.QueryString(field_identifier & "val1")
End Function
%>
```

Figure 154:Custom-Field-Type Properties Dialog Box—ASP Form Tab

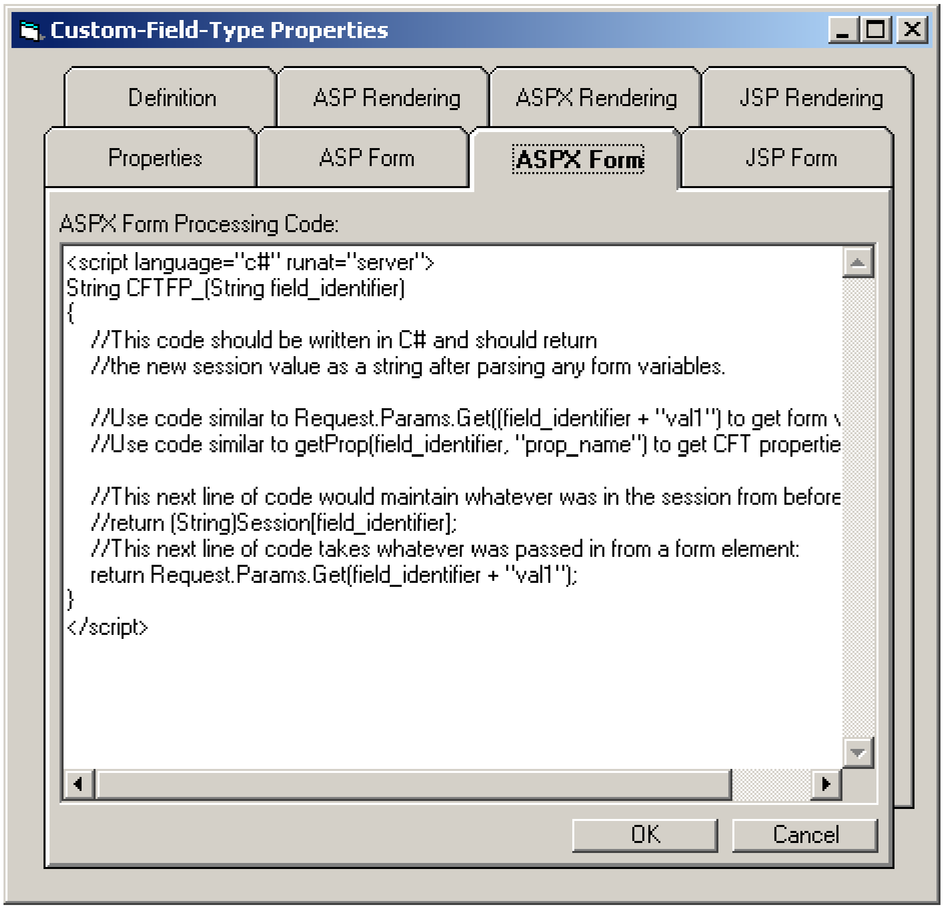
The function shown in Figure 154 takes the form value, and adds the value of the `add_to_end` property to the end of the session variable.

Properties defined on the `Properties` tab can be referenced in the ASP Form Process code. To do this, the property is referenced by the `getProp()` function call.

ASPX Form Tab

The ASPX Form Process code takes what is submitted by the form and returns the session variable with its new value.

The ASPX Form Code is C# code that takes in the field identifier as an argument, and returns a new value for the session variable. Figure 155 shows an example of code that might be entered on the `ASPX Form` tab.



ASPX Form Processing Code:

```
<script language="c#" runat="server">
String CFTFP_(String field_identifier)
{
   //This code should be written in C# and should return
   //the new session value as a string after parsing any form variables.

   //Use code similar to Request.Params.Get((field_identifier + "val1") to get form v
   //Use code similar to getProp(field_identifier, "prop_name") to get CFT propertie

   //This next line of code would maintain whatever was in the session from before
   //return (String)Session[field_identifier];
   //This next line of code takes whatever was passed in from a form element:
   return Request.Params.Get(field_identifier + "val1");
}
</script>
```

Figure 155:Custom-Field-Type Properties Dialog Box—ASPX Form Tab

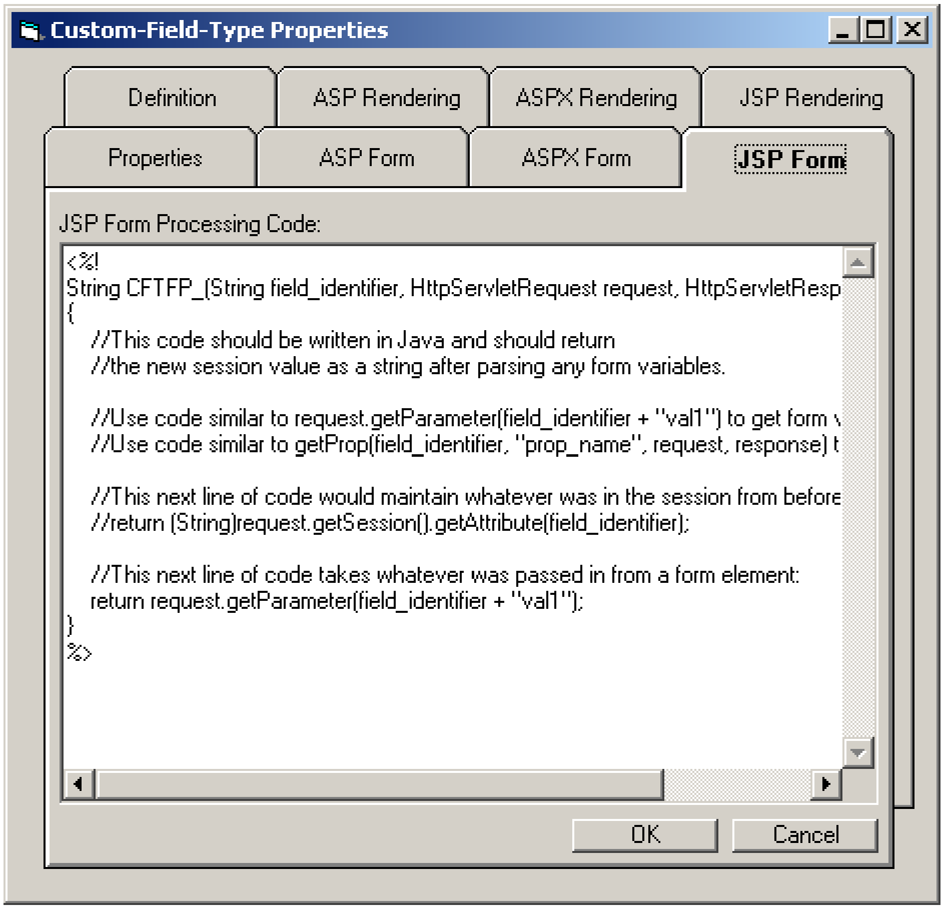
The function shown in Figure 155 takes the form value, and adds the value of the `add_to_end` property to the end of the session variable.

Properties defined on the `Properties` tab can be referenced in the ASPX Form Process code. To do this, the property is referenced by the `getProp()` function call.

JSP Form Tab

The JSP Form Process code takes what is submitted by the form and returns the session variable with its new value.

The JSP Form Code is Java code that takes in as arguments the field identifier, HttpServletRequest, and HttpServletResponse, and returns a new value for the session variable. Figure 156 shows an example of code that might be entered on the `JSP Form` tab.



JSP Form Processing Code:

```
<%!
String CFTFP_(String field_identifier, HttpServletRequest request, HttpServletResp
{
   //This code should be written in Java and should return
   //the new session value as a string after parsing any form variables.

   //Use code similar to request.getParameter(field_identifier + "val1") to get form v
   //Use code similar to getProp(field_identifier, "prop_name", request, response) t

   //This next line of code would maintain whatever was in the session from before
   //return (String)request.getSession().getAttribute(field_identifier);

   //This next line of code takes whatever was passed in from a form element:
   return request.getParameter(field_identifier + "val1");
}
%>
```

Figure 156:Custom-Field-Type Properties Dialog Box—JSP Form Tab

The function shown in Figure 156 takes the form value, and adds the value of the `add_to_end` property to the end of the session variable.

Properties defined on the `Properties` tab can be referenced in the JSP Form Process code. To do this, the property is referenced by the `getProp()` function call.

Creating a Stored Procedure

A *Stored Procedure* is a program that is physically stored within a database. The advantage of a Stored Procedure is that it is run directly by the database engine, and it is generally faster at processing database requests. The database server has direct access to the data it needs to manipulate, and it needs to send only the final results back to the user, thus eliminating the overhead of communicating potentially large amounts of interim data back and forth.

To create a database Stored Procedure, you must create a Database Interface. The Database Interface defines the Stored Procedure to call, along with any input, output, and return parameters that the Stored Procedure requires. For more information about creating a Database Interface, refer to “Creating a Database Interface” on page 178.

In order to execute the Database Interface, you need to create an Action that references the Database Interface. This Action is automatically created when the Database Interface is saved, if you previously configured the settings to do so by using the File menu (File > Settings > Automatically Create Actions for New Interfaces). Otherwise, you will have to create an Action, and then manually add the call to the Database Interface, in the code section for the Action. Once the Action has been created you can add it to any page.

Procedure: Create a Database Interface that will be used to call your Stored Procedure

Start of procedure

1. Select Define > Database Interfaces to open the Database Interface List dialog box.
2. Click Add to open the Database Interface Properties dialog box.
3. Specify information for the following fields:
   - Interface—Specify a name for the Database Interface. Any Actions that execute this Interface will reference it by this name.
   - Database—Use the drop-down list to specify the database that contains the Stored Procedure that you want to call.
   - Query Type—Use the drop-down list to select Stored Procedure.

End of procedure

Once the initial Database Interface properties have been set, you can construct the Action to call a Stored Procedure. This Action is referred to as a *Query*.

Procedure: Add a Query

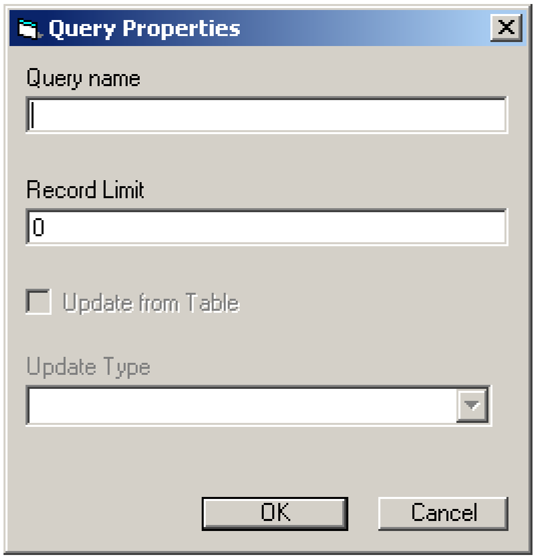
Start of procedure

1. Click Add Query.

   The following message will appear:

   This interface must be saved to continue. Save?
2. Click Yes.

   The Query Properties dialog box opens (see Figure 157).



Query name

Record Limit
0

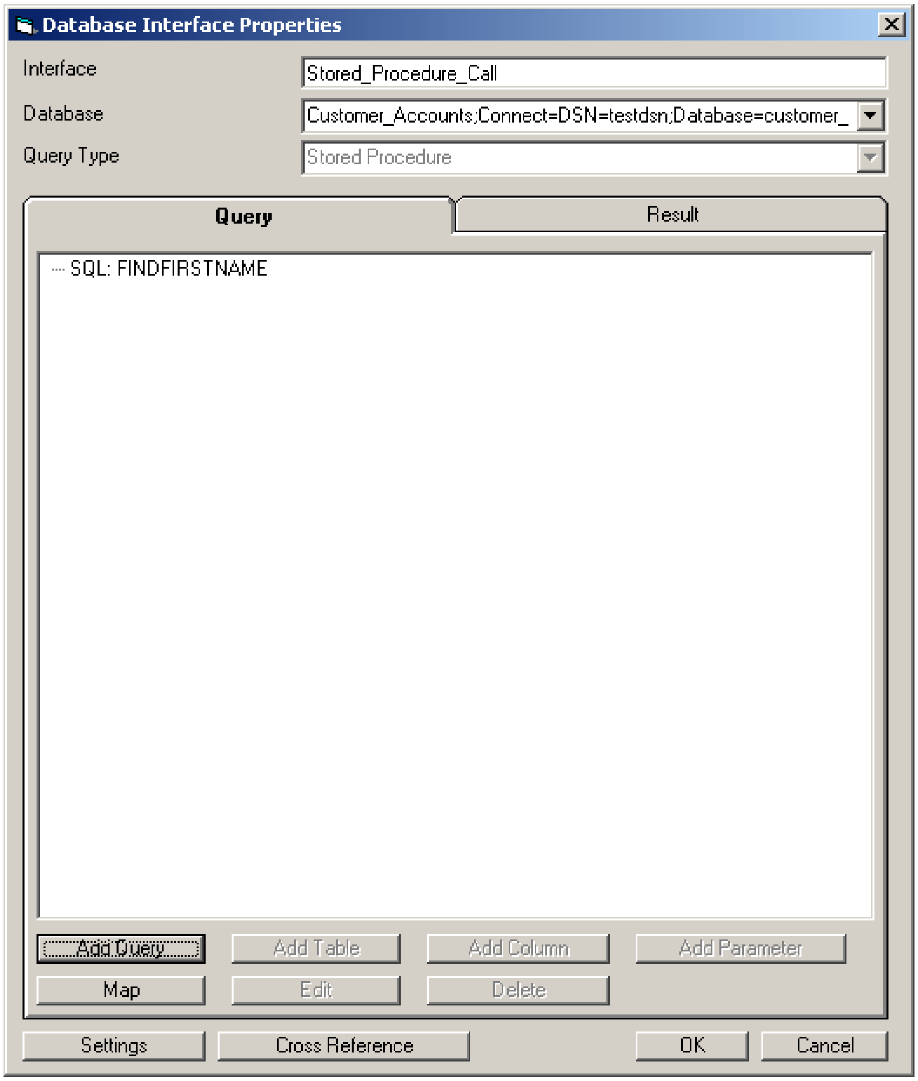
□ Update from Table

Update Type

Figure 157:Query Properties Dialog Box

3. In the `Query Name` text box, enter the name of the Stored Procedure as it exists in the database, and then click `OK`.

   An SQL element representing the query is added to the `Query` tab of the `Database Interface Properties` dialog box as shown in Figure 158.



Database Interface Properties

Interface: Stored_Procedure_Call

Database: Customer_Accounts;Connect=DSN=testdsn;Database=customer_

Query Type: Stored Procedure

SQL: FINDFIRSTNAME

Figure 158:Database Interface Properties Dialog Box—Query Tab

End of procedure

You must now define the parameters that the Stored Procedure requires.

A parameter must be defined for each parameter that the Stored Procedure contains, and the parameters must be defined in the order in which they are specified in the Stored Procedure. If the Stored Procedure returns a value, the first defined parameter must be for the returned value.

The following is an example of a Stored Procedure:

```
CREATE OR REPLACE PROCEDURE FINDFIRSTNAME(INCUSTID IN NUMBER,
FirstName OUT VARCHAR2)
IS

BEGIN
```

Select FName

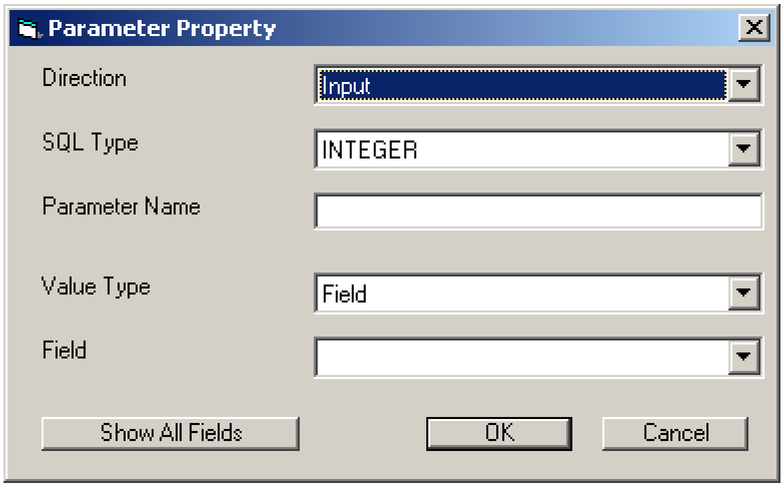
```
INTO FirstName
From CUSTOMER
Where CUSTID = INCUSTID;
COMMIT;

END;
```

Procedure:
Define parameters that will match your Stored Procedure requirements

Start of procedure

1. On the `Query` tab of the `Database Interface Properties` dialog box, select the SQL element, and then click `Add Parameter`.

   The `Parameter Property` dialog box opens, as shown in Figure 159.



Parameter Property

Direction: Input

SQL Type: INTEGER

Parameter Name:

Value Type: Field

Field:

Figure 159:Parameter Property Dialog Box

Table 72 describes the options on the `Parameter Property` dialog box.

Table 72: Custom Field Type Property Options

Control	Description / What to Do...
Direction	A drop-down list containing the direction of the parameter. The direction must match the direction of the corresponding parameter in the Stored Procedure. You can select one of the following values from the drop-down list: • Input • Output • Input/Output • Return Value
SQL Type	A drop-down list containing the SQL type of the parameter. It must match the type of the corresponding parameter in the Stored Procedure. You can select one of the following values from the drop-down list: • INTEGER • SMALLINT • BIGINT • DOUBLE • REAL • DECIMAL • FLOAT • VARCHAR • DATE • TIME • TIMESTAMP **Note:** Genesys Agent Scripting partially supports mapping the DB2 BIGINT data type in the Database Interface. This data type is supported with the following exceptions. • Cannot create a Database Interface with a Where clause referencing a BIGINT column. • Cannot create a Database Interface that performs an Insert or Update specifying a value for a BIGINT column.
Parameter Name	Name of the parameter specified in the Stored Procedure.

Table 72: Custom Field Type Property Options (Continued)

Control	Description / What to Do...
Value Type	A drop-down list containing the type of the value. • If the direction is Input, this field defines whether the input value of the parameter is a specific value or the contents of a specific field. • If the direction is Output or Return Value, this field must be set to Field, in order to define which field is to contain the output/return parameter value. This field is disabled for a parameter whose SQL Type is set to CURSOR. An output CURSOR parameter can be saved to a Table field by adding a Database Table and its associate columns to the Database Interface, matching results from the cursor, and mapping it to a Genesys Agent Scripting Field of type Table.
Field	Use the drop-down box to select either the field whose content is to be passed as an Input parameter, or the field that is to contain the value of an Output/Return Value parameter. This option appears only if the Value Type is set to Field.
Value	Use this field to specify a static value for an Input parameter. This option appears only if Value Type is set to Value.
Show All Fields	Changes the drop-down list so that it contains all Genesys Agent Scripting fields.

2. Once all your parameters have been added, click OK to return to the Database Interface List dialog box.

A Store Procedure can return query data from an SQL Select referred to as a *ResultSet*. For most databases, such as SQL Server, this ResultSet can be returned automatically, without having to define a parameter for it. For the Oracle database, however, a Stored Procedure can return the ResultSet only through an output cursor parameter. In both cases, this ResultSet needs to be mapped to an Agent Scripting Table Field. This is accomplished by defining a Database table with columns that match those from the ResultSet. The defined Database table can then be mapped to an Agent Scripting Table Field. For information on how to map a Database table to a Genesys Agent Scripting Table Field, refer to “Using the Map Button” on page 198.

Although Genesys Agent Scripting supports calls to Stored Procedures, in general the following restrictions apply:

- Input cursor parameters are not supported.
- Only one output cursor parameter can be defined.
- For Database providers `System.Data.SqlClient` and `System.Data.OracleClient`, parameters must be defined with parameter names.
- For Database provider `System.Data.OracleClient`, a returned cursor parameter is not supported.
- For OLEDB access to Oracle versions earlier than 9.2.0.4.0, timestamp parameters are not supported.

End of procedure

Importing Objects from Other Project Books

The `Import` feature of Genesys Agent Scripting allows you to import objects from other Project Books into the current Project Book. This is useful if you have defined Actions, Fields, Database Interfaces, Catalogs, Page Layouts, or other objects in a different Project Book and would like to make those objects available in the Project Book you are currently working on.

Import operations are performed on a hierarchical basis, so that if an object is selected to be imported, all other objects required by that object will be imported as well. For example, if a Page is selected to be imported, all Fields and Actions used by that Page will automatically be selected to be imported.

You import objects from one Project Book to another using the `File` menu.

Procedure:
Import Objects from One Project Book to Another

Start of procedure

1. Select `File` > `Import` to open the `Import From Project Book` dialog box as shown in Figure 160.

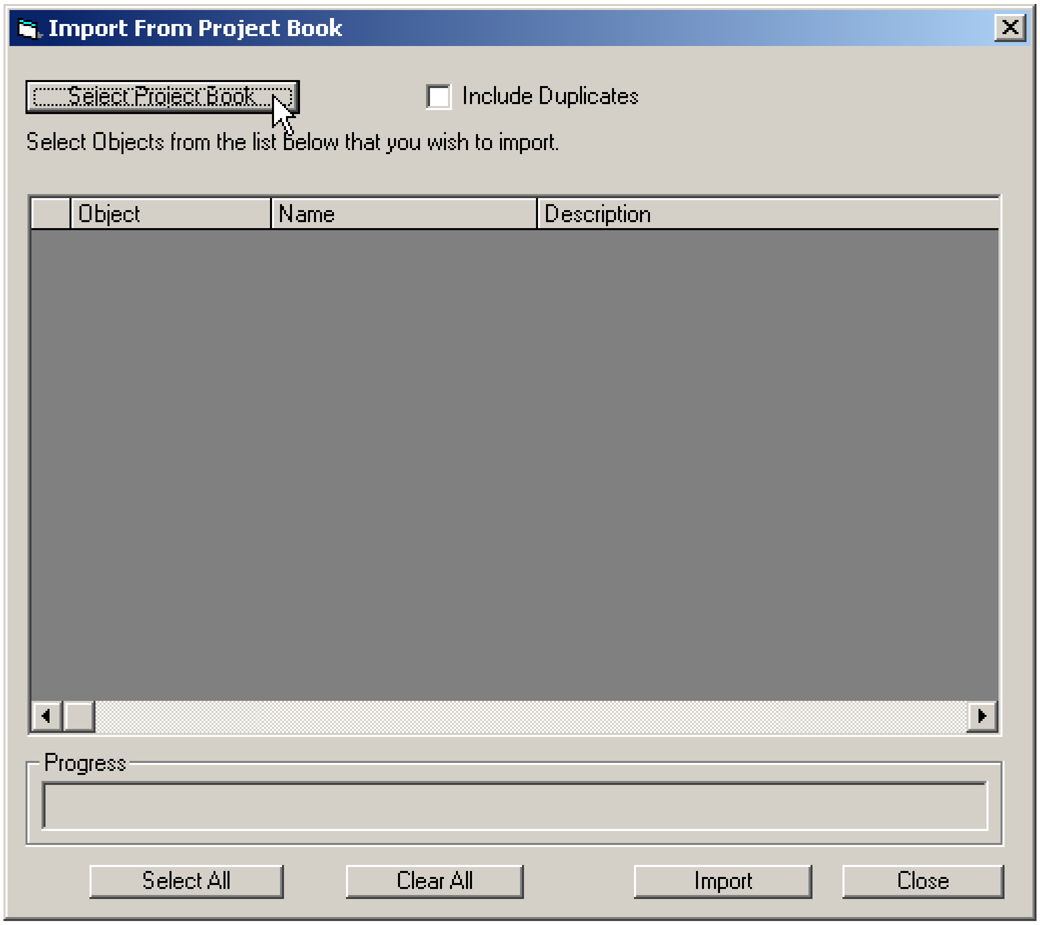


Figure 160:Import From Project Book Dialog Box

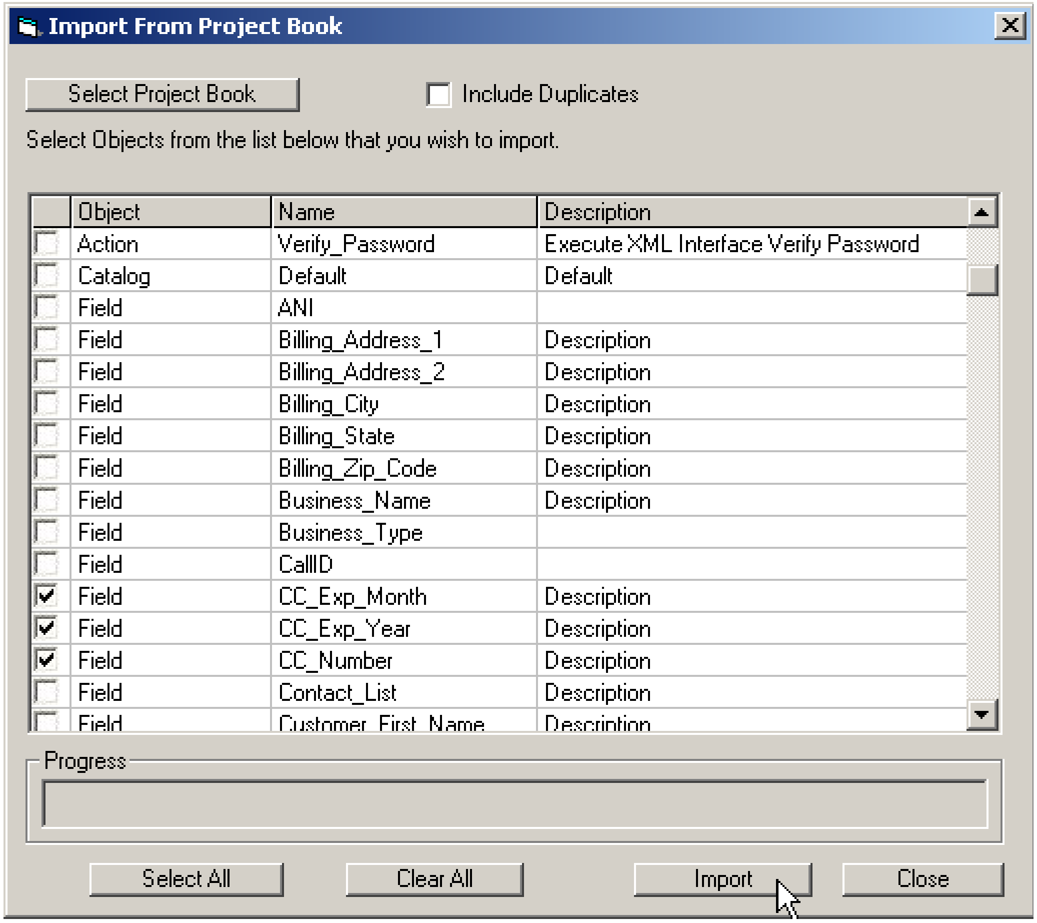
2. Select the Project Book and the objects you wish to import at the `Import From Project Book` dialog box.

   Table 73 describes the options on the `Page Properties` dialog box.

Table 73: Import From Project Book

Control	Description / What to Do...
Select Project Book	Click `Select Project Book` to open a dialog box from which you can select or navigate to a Genesys Agent Scripting Project Book (`.mdb`) file.
Include Duplicates	If you select this check box, the `Object Table` will display all objects in the selected Project Book whether or not they are already included in the currently open Project Book. If you leave this check box unselected, only objects that aren't in the currently open Project Book will be displayed.
Object Table (Object/Name/Description)	After you select the Project Book, this table will display a list of objects that you may select to be imported. Select the check box beside each object to select it. An example is shown in Figure 161 on page 340.
Progress	Once you click `Import,` the Progress bar shows the progress of the import process.
Select All	Click `Select All` to select all of the objects in the `Object Table.` This places a check in the check box beside each object.
Clear All	Click `Clear All` to clear any objects in the `Object Table` that have been selected.
Import	Click `Import` to start the process of importing the selected objects into the currently open Genesys Agent Scripting Project Book.
Close	Click `Close` to close the `Import From Project Book` dialog box.

3. Select objects in the Import from Project Book dialog box to import after you have selected a Project Book, as shown in Figure 161.



Select Objects from the list below that you wish to import.

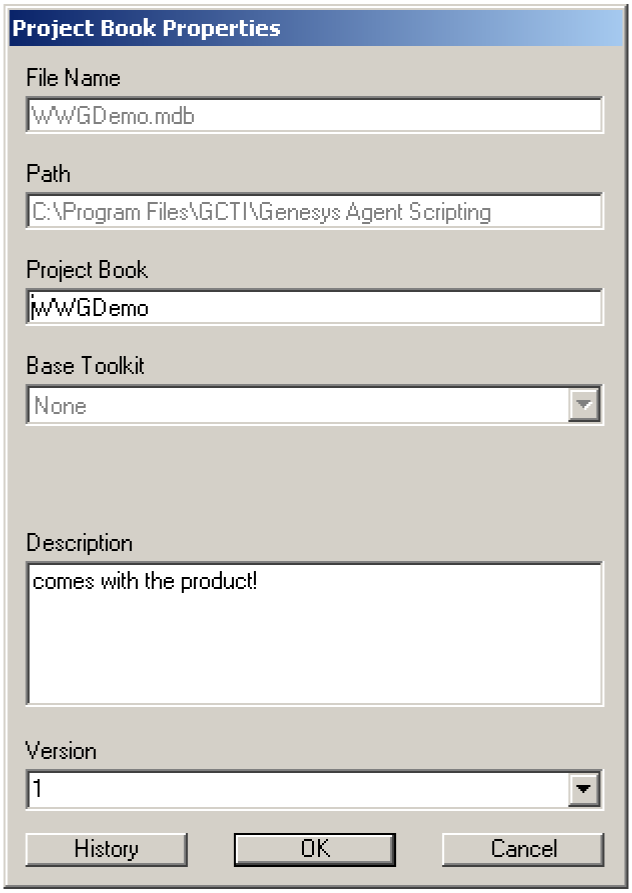
	Object	Name	Description
□	Action	Verify_Password	Execute XML Interface Verify Password
□	Catalog	Default	Default
□	Field	ANI	
□	Field	Billing_Address_1	Description
□	Field	Billing_Address_2	Description
□	Field	Billing_City	Description
□	Field	Billing_State	Description
□	Field	Billing_Zip_Code	Description
□	Field	Business_Name	Description
□	Field	Business_Type	
□	Field	CallID	
☑	Field	CC_Exp_Month	Description
☑	Field	CC_Exp_Year	Description
☑	Field	CC_Number	Description
□	Field	Contact_List	Description
□	Field	Customer_First_Name	Description

Figure 161:Import From Project Book Dialog Box - Select Objects

End of procedure

Managing Multiple Versions of Scripts

Genesys Agent Scripting provides a Field called `Version` in the Project Book Properties dialog box, as shown in Figure 162. You can enter any alphanumeric characters to make it meaningful for keeping track of your scripting project.



Project Book Properties

File Name: WWGDemo.mdb

Path: C:\Program Files\GCTI\Genesys Agent Scripting

Project Book: WWGDemo

Base Toolkit: None

Description: comes with the product!

Version: 1

Figure 162:Project Book Properties - Version Field

For example, there is an existing demo Project Book called `WWGDemo`, which has a Version `1` and a filename `WWGDemo.mdb`.

Whenever changes might be required for the `WWGDemo` Project Book, for example, you can either change the original version or *make a copy of it and change that*.

Procedure:
Copy the original version of the Project Book

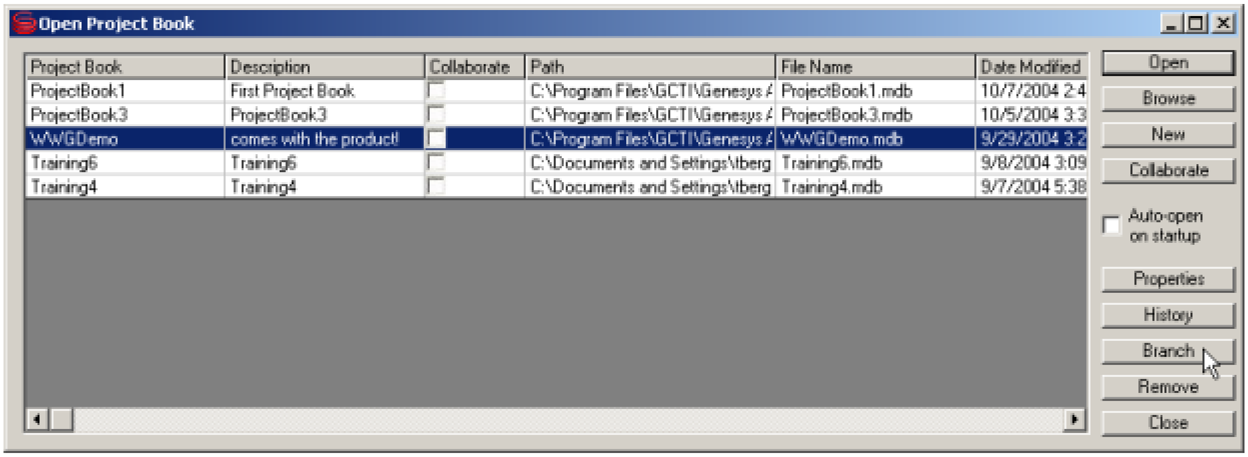
Purpose: To safely make changes on the copy.

Start of procedure

1. Select `File` > `Open Project Book`.

The `Open Project Book` dialog box opens as shown in Figure 163.

2. Select the project `WWGDemo` and click the `Branch` button.



Project Book	Description	Collaborate	Path	File Name	Date Modified
ProjectBook1	First Project Book	☐	C:\Program Files\GCTI\Genesys A	ProjectBook1.mdb	10/7/2004 2:4
ProjectBook3	ProjectBook3	☐	C:\Program Files\GCTI\Genesys A	ProjectBook3.mdb	10/5/2004 3:3
WWGDemo	comes with the product!	☐	C:\Program Files\GCTI\Genesys A	WWGDemo.mdb	9/29/2004 3:2
Training6	Training6	☐	C:\Documents and Settings\tberg	Training6.mdb	9/8/2004 3:09
Training4	Training4	☐	C:\Documents and Settings\tberg	Training4.mdb	9/7/2004 5:38

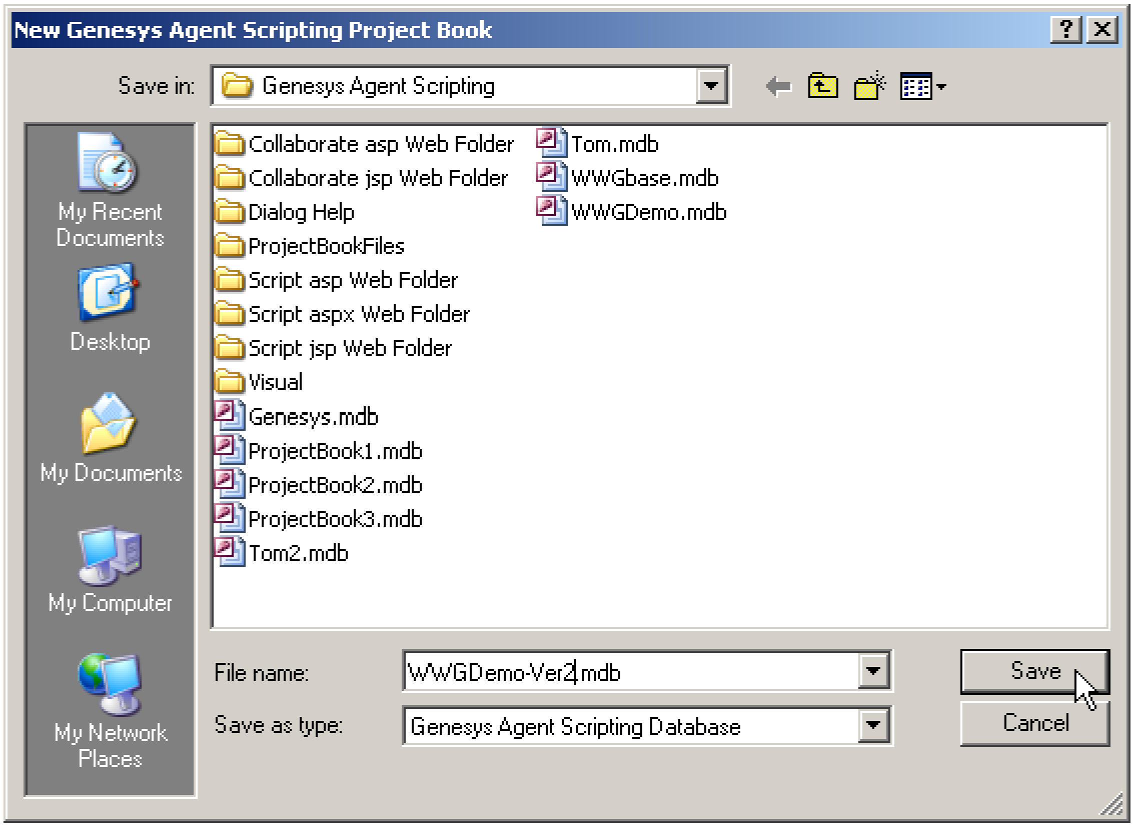
☐ Auto-open on startup

Figure 163:Open Project Book Dialog Box - Click Branch

The New Genesys Agent Scripting Project Book dialog box opens as shown in Figure 164. The File Name dropdown list shows the original Project Book name with the word Branch added.

3. Change the file name to something meaningful for your future branched version.

Note: The name cannot be the same as the original file name in the same path or the original file will be overwritten.



New Genesys Agent Scripting Project Book

Save in: Genesys Agent Scripting

File name: WWGDemo-Ver2.mdb

Save as type: Genesys Agent Scripting Database

Figure 164:New Genesys Agent Scripting Project Book - New Name

4. Click `Save` to save the copy of your Project Book under the new name.

   The `Project Book Properties` dialog box opens again with the `Project Book` field containing a default of the original Project Book name appended with `(branch)`.

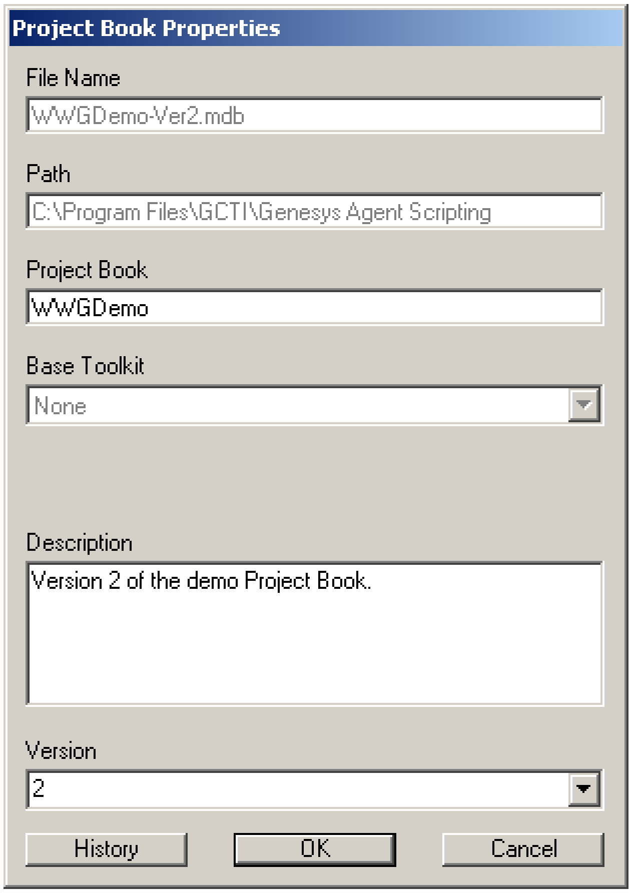
5. Change the name of the Project Book to something more meaningful.

   For example, since this is a *new version* of the same Project Book, you can continue to use the original Project Book name (`WWGDemo` in our example).

6. Change the description to give more information about this branched version if you choose.

7. Next, change the version by typing a new version number in the `Version` field.

   Figure 165 shows an example illustrating all these changes.



Project Book Properties

File Name: WWGDemo-Ver2.mdb

Path: C:\Program Files\GCTI\Genesys Agent Scripting

Project Book: WWGDemo

Base Toolkit: None

Description: Version 2 of the demo Project Book.

Version: 2

Figure 165:Project Book Properties - Changed to Version 2

8. Click `Close` to return to the Genesys Agent Scripting Development Environment.

Changes that you make at this point will affect the new version of the Project Book only. If you followed the steps above, the earlier version of the Project Book will remain unaffected.

9. Repeat the entire process described in this section when you wish to create yet another separate version of the Project Book.

End of procedure

Examples

This section shows step-by-step examples that illustrate the features described in this chapter:

- Creating a new Catalog
- Creating a new Page Layout and adding a Catalog to it

- Importing objects from the demo Project Book (WWGDemo.mdb)

Creating a New Catalog

Let's create a new Catalog for the WWGDemo Project Book that displays only the Pages of the Outbound_Customer_Alarm Stream.

Procedure: Create a New Catalog

Start of procedure

1. Open the WWGDemo Project Book.
2. Select Define > Catalogs to open the Catalog List dialog box.
3. Click Add to initiate the process for creating a *new* Catalog.
4. In the Catalog Properties dialog box, name the Catalog Outbound.
5. In Process Flows, select Outbound_Customer_Alarm.
6. In Streams, select *and hold* Outbound_Customer_Alarm, then drag it into the Catalog area.
7. In the Catalog dropdown list, select Show Streams, and click Yes at the message box.
8. Select Include Current Page Indicator, and click OK.
9. Your new Catalog is added to the Catalog List.
10. Click Close.

End of procedure

Creating a New Page Layout and Add a Catalog

Procedure: Create a new Page Layout

Purpose: To create a new Page Layout in the WWGDemo Project Book, add the Outbound Catalog we just created, and make some minor adjustments to the appearance of the Page Layout.

Start of procedure

1. Select File > Page Layouts > Insert New Page Layout to open the Page Layout dialog box.
2. Type a name (such as NewLayout) for this Page Layout in the Layout Name field.
3. In the Catalog dropdown list, select the Outbound Catalog you created in the example above.
4. Click OK to complete the Page Layout and close the dialog box.
5. The Page Layout is created. You can edit it using the File menu to select the Page Layout by name.
6. Select File > Page Layouts > NewLayout (or other name you typed), to open the Page Layout Editor.
7. Select and delete the Footer area.
8. Select and move the Catalog area to where the Footer area used to be.
9. Select and stretch the Page area to the left where the Catalog used to be.

   You now have a Header area, Page area, and Catalog area from top to bottom in the Page Layout.

   You should now assign this Page Layout to an existing Page.
10. Click any Page in the Page Tree, then select Page > Edit Page to open the Page Properties dialog box.
11. Select your new Page Layout from the Layout dropdown list and click OK to assign the new Page Layout to this Page.

End of procedure

Compile and Test

When you have a script segment that you wish to test, such as the example we've just completed above, you should compile the code and run Simulate to see how things work.

The procedures for compiling and testing (simulating) your code are described in Chapter 12, "Compiling, Testing, and Deploying Your Application," on page 269.

Import Objects From the WWGDemo Project Book

Procedure:
Import objects into Another Project Book

Purpose: Import some objects from the demo application WWGDemo into another Project Book.

Start of procedure

1. Open any Project Book other than the WWGDemo Project Book, or create a new Project Book as described in Chapter 3, “Creating a New Project Book,” on page 51
2. Select File > Import to open the Import From Project Book dialog box.
3. Click Select Project Book, find and select WWGDemo.mdb, and click OK.

   The Object Table is populated with all objects from WWGDemo that may be imported.
4. Select any number of objects to import.

   These can be Actions, Fields, Streams, Pages, Page Layouts, Catalogs, or any other objects that you want to have available in your current Project Book.
5. Click Import to import the selected objects.

You will find your imported Actions on the Action List, imported Fields on the Field List, imported Pages on the Page Tree, imported Page Layouts in the File menu, imported Catalogs in the Catalog List when you select Define > Catalogs, and so on. In short, the imported objects are available in your current Project Book where you would expect to see them had you defined or created them yourself.

End of procedure

Chapter

14 Toolkits

This chapter provides an overview of the toolkits that can be used when creating a new Project Book. It also introduces the Reporting functionality.

The information in this chapter is divided among the following topics:

Overview

Genesys Agent Scripting now allows you to create a new Project Book containing pre-defined parts such as Fields, Actions, and Pages, to provide access to new features. When creating a new Project Book, you can select one of the following toolkits:

- Agent Scripting Toolkit
- Genesys Agent Interaction Toolkit

If you want to upgrade an existing Project Book with one of the toolkits, then perform the following steps

- Create the new Project Book, selecting the desired toolkit.
- Import all parts from the existing Project Book into the new one.

Note: You can not just import the toolkit into an existing Project Book. The Project Book is marked with a toolkit when it is created.

Both toolkits provide the following features

- Suspend a script and resume it later
- Report to a database changes to fields

In addition, the Genesys Agent Interaction Toolkit interfaces with the Genesys Framework, providing the following features:

- Query agent information
- Query and modify outbound records
- Query and modify contacts
- Query and modify interaction attached data
- Disconnect an interaction
- Mark done an interaction
- Transfer (one-step and two-step) an interaction

When a Project Book is created with the Genesys Agent Interaction Toolkit, the Advanced Settings option `Integrate with Genesys Agent Interaction` is enabled and automatically set. If set, then the Genesys Agent Scripting generated web application will contain the following features:

- On any Page refresh, all context (Field information) and the Page stack are saved in memory, using the Interaction ID as the key.
- When a script is launched, an attempt is made to reload the context and Page stack from memory for the provided Interaction ID.
- When a script is launched, information regarding the agent, interaction, contact, and outbound are retrieved from the Genesys Framework.

This Advanced Setting option is useful when integrating the generated web application with the Genesys Agent Desktop (GAD), where the web application is displayed within a GAD frame. GAD always relaunches the web application whenever a GAD user tabs into the frame. This forces the web application to start over with the first Page in the script, losing all Field information previously entered. With this Advanced Setting, the generated web application will reload the information that was previously saved when the GAD user left the frame. This setting is optional since this behavior may not be desirable when the generated web application is not going to be launched multiple times for the same interaction.

A sample Project Book is provided showing some of the features of the Genesys Agent Interaction Toolkit. The Project Book is located in the Agent Scripting install directory underneath `GenesysIntegration`. The name of the Project Book is `GenesysDemo`.

Configuration

The following section provides configuration information for the fields that are provided within the Agent Scripting Toolkit and Genesys Agent Interaction Toolkit.

The configuration fields listed below will appear in all Target Environments created for the Project Book, on the `Configuration File` tab. This is where you initialize the configuration fields.

For the Agent Scripting Toolkit, the available fields are:

- `G_System_DB_Connection_String.`
- `G_System_DB_Provider.`

For the Genesys Agent Interaction Toolkit, the available fields are:

- `G_Config_ApplName.`
- `G_Config_BackupHost.`
- G_Config_BackupPort.
- `G_Config_PrimaryHost.`
- `G_Config_PrimaryPort.`
- `G_System_DB_Connection_String.`
- `G_System_DB_Provider.`
- `XMLServerURL.`
- `G_NETServer_PrimaryHost.`
- `G_NETServer_PrimaryPort.`
- `G_NETServer_BackupHost.`
- `G_NETServer_BackupPort.`

Note: The items in bold above are Target Environment specific configuration fields.

Database Configuration

To support the features for reporting and suspend/resume, a database must be configured to store the information. This database can be DB2, Oracle, or Microsoft SQL Server.

Table Creation

To create the required tables, you can run one of the SQL files that resides in the default install folder:

`C:\Program Files\GCTI\Genesys Agent Scripting\GenesysIntegration\sql`

The SQL files are:

`create_tables_db2.sql`. Creates required tables for a DB2 database `create_tables_oracle.sql`. Creates required tables for an Oracle database

`create_tables_mssql.sql`. Creates required tables for a Microsoft SQL Server database

Database Connection

The toolkits provide the following configuration fields to establish a connection to the Database:

`G_System_DB_Provider`. A dropdown list of the supported Database providers (for example, `JDBC`, `OLEDB`, or an `ASP.NET` Manager Provider)

`G_System_DB_Connection_String`. A connection string to connect to the database for the selected Database provider.

JSP Target Environment (AIL) Configuration

Configuration Server Connection

The Genesys Agent Interaction Toolkit provides the following additional configuration fields to establish a connection to the Genesys Configuration Server:

`G_Config_PrimaryHost`. (Required) The host name of the primary Genesys Configuration Server

`G_Config_PrimaryPort`. (Required) The port of the primary Genesys Configuration Server

`G_Config_BackupHost`. (Optional) The host name of the backup Genesys Configuration Server

`G_Config_BackupPort`. (Optional) The port of the backup Genesys Configuration Server

`G_Config_AppName`. Name of the application created with the template `Genesys_Agent_Scripting_720`. An application must have been previously configured with this template using the Genesys Configuration Manager.

ASP .NET Target Environment (Genesys Integration Server) Configuration

In order to compile a Project Book for the ASPX Target Environment, you must enter appropriate values into the configuration fields that describe information about Genesys Integration Server, and that are to be used by ASPX Agent scripts.

Genesys Integration Server Connection

The following configuration fields were added to the `Configuration File` tab for ASPX Target Environments (`Compile > Target Environment`). They are used instead of the JSP AIL Target Environment configuration fields:

`G_NETServer_PrimaryHost`. (Required) The host name of the primary Genesys Integration Server

`G_NETServer_PrimaryPort`. (Required) The port of the primary Genesys Integration Server

`G_NETServer_BackupHost`. (Optional) The host name of the backup Genesys Integration Server

`G_NETServer_BackupPort`. (Optional) The port of the backup Genesys Integration Server

Input Parameters

To launch a Genesys Agent Scripting generated Web Application, a set of input parameters can be specified in the URL. The following input parameters are valid for those Project Books created with the Genesys Agent Interaction Toolkit:

`G_Assigner_URL`. (Optional) The URL address where the Genesys Agent Scripting Assigner Servlet resides. This parameter is required if the generated web application is integrated with the Genesys Agent Desktop application.

`G_Script_Query_Relationship`. (Optional) A `true` (`0`) or `false` (`1`) value indicating whether to automatically branch to the script Page if no Script ID was provided but a relationship ID was provided.

`G_Script_RelationshipID`. (Optional) Relationship ID.

`G_Config_Place`. (Required) The Place where the agent resides. This was previously configured with the Genesys Configuration Manager.

`G_Agent_ID`. (Required) The ID of the agent. This was previously configured with the Genesys Configuration Manager.

`G_Script_ID`. (Optional) Script ID. If provided, Agent Scripting will attempt to resume the suspended script instance identified by the Script ID. If not specified, then a Script ID value will be derived by either resuming another script instance or generating a new Script ID.

`G_Interaction_ID`. (Optional) Interaction ID value. The value can either be from the T-Server (for example, a numeric value) or from the Agent Interaction Layer (for example, `Phonecall-0`).

`G_Contact_ID`. (Optional) Contact ID value from the Contact Server.

`G_OR_ID`. (Optional) Outbound Record Handle ID value from the Outbound Contact Server.

Action Response Fields

Both toolkits contain Actions that take Fields as input and output parameters. All toolkit Actions return their results in the following general response Fields:

G_System_Response_Result. Numeric result of the Action. A value of 0 indicates success.

G_System_Response_Description. A general message indicating the success or failure of the Action.

G_System_Response_Detail. If G_System_Response_Result contains a non-zero value, then this Field will contain a detailed error message.

Suspending and Resuming Scripts

Genesys Agent Scripting provides a set of pre-defined Fields, Actions, and Pages that allows a script designer to create scripts with the ability to suspend them and resume them later. These pre-defined elements are automatically copied into any new Project Book when the Project Book is configured with one of the two supported base toolkits (see Step 4 in the section "Creating the Project Book" on page 52).

Suspend

The agent who navigates through a series of Pages can now suspend the script, saving to a database the information on all the Pages navigated. The Agent can then later resume the script which will automatically navigate to the Page that was displayed at the time when the script was suspended. All Field information displayed on all Pages is restored. The Agent who suspended the script does not have to be the same Agent who can later resume it.

When a script is started, an ID is generated to uniquely identify the instance of the script. The agent can suspend the instance of the script by executing the action G_Script_Suspend. After the script has been suspended, the agent is brought to the Process Flow/Stream/Page Genesys/Genesys_Action/G_Script_Suspend, which allows the agent to enter a relationship ID and a description for the suspended script. The relationship ID can help identify those suspended scripts associated with a particular customer. The description can help identify why a specific script instance was suspended.

The following lists the Fields and Actions on the G_Script_Suspend Page.

Field Definitions

G_Script_ID. The ID of the script that was suspended.

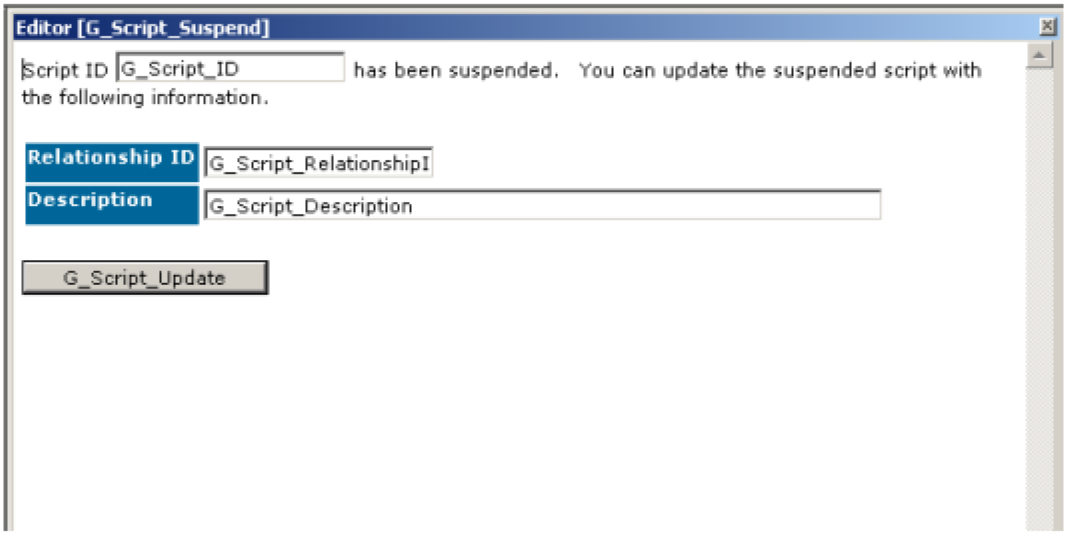
G_Script_RelationshipID. The relationship ID with which the agent wants to tag the suspended script.

G_Script_Description. The description with which the agent wants to tag the suspended script.

Action Definitions

G_Script_Update. This Action tags the suspended script with the supplied relationship ID and description.

Figure 166 shows an example of the G_Script_Suspend Page.



Editor [G_Script_Suspend]

Script ID G_Script_ID has been suspended. You can update the suspended script with the following information.

Relationship ID	G_Script_RelationshipI
Description	G_Script_Description

Figure 166:Example of G_Script_Suspend Page

Resume

When the agent wants to resume a suspended script, the agent navigates to the Process Flow/Stream/Page Genesys/Genesys_Scripts/G_System_Scripts. This Page provides the ability to display all suspended scripts in the system or to display only those script IDs associated with a specific relationship ID. The returned list is displayed in a table with each entry containing the description text entered when the script was suspended. The agent selects the script from the table and executes the action G_Script_Resume to resume the script.

The following lists the Fields and Actions on the G_System_Scripts Page.

Field Definitions

G_Script_RelationshipID_Selected_Entry. This field represents a relationship ID whose value is input to the G_Script_GetRelated_Scripts Action, which is used to query for suspended scripts with a specific relationship ID.

G_Script_Table. This field contains a table to hold the results of the query for suspended scripts. It contains the following Field columns:

G_Script_ID_Selected_Entry. Uniquely identifies the instance of a suspended script. The ID was created when the instance of the script was started.

G_Script_RelationshipID_Selected_Entry. The ID that is supplied by the agent when the script was suspended. This ID can show a relationship between more than one suspended scripts.

G_Script_Version. The version of the Project Book.

G_Script_FlowName. The name of the Process Flow that was being suspended.

G_Script_StartTime. The time at which the instance of the script was started.

G_Script_SuspendTime. The time at which the instance of the script was suspended.

G_Script_Description. The description of the suspended script supplied by the agent when the script was suspended. This helps the agent to identify which script to resume.

Action Definitions

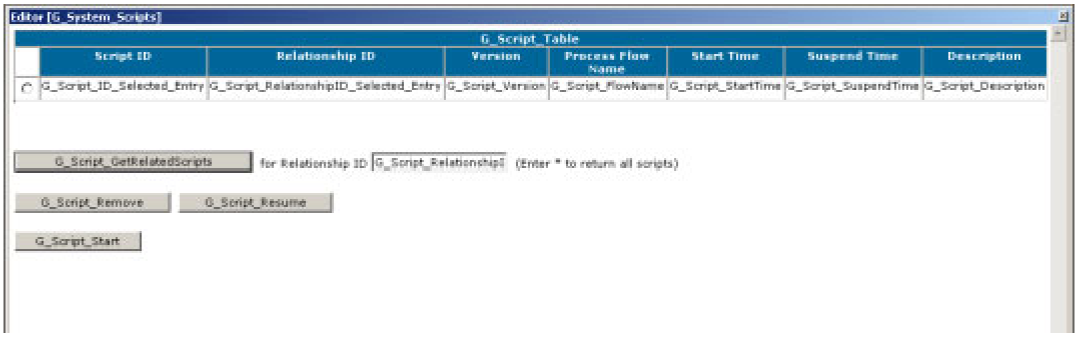
G_Script_Resume. This Action resumes the script selected from the script table. The agent is navigated to the instance of the selected script.

G_Script_Remove. This Action removes from the system the script that was selected from the script table.

G_Script_Start. This Action creates a unique ID for the current instance of the script. It is only displayed if a script ID was never generated when the instance of the script was first started.

G_Script_GetRelated_Scripts. This Action queries the database for suspended scripts. You can query the database for all suspended scripts or only those scripts containing a specific relationship ID.

Figure 167 shows an example of the G_System_Scripts Page.



G_Script_Table

	Script ID	Relationship ID	Version	Process Flow Name	Start Time	Suspend Time	Description
○	G_Script_ID_Selected_Entry	G_Script_RelationshipID_Selected_Entry	G_Script_Version	G_Script_FlowName	G_Script_StartTime	G_Script_SuspendTime	G_Script_Description

Figure 167:Example of G_System_Scripts Page

The supported Actions listed above will set the following Fields to reflect the response of the Action:

G_System_Response_Result. Returns a numeric string value. A value of 0 indicates that the Action was successful.

G_System_Response_Description. Returns a description of the result

G_System_Response_Detail. Returns detail information if the result was not successful (for example, exception text).

Note: The Pages G_System_Suspend and G_System_Scripts do not display any of the response Fields. You can add these response Fields to the Pages, or you can add them once to the defined Page Layout (for example, in the Footer section).

Script Start Scenarios

As mentioned earlier, the suspend/resume functions are provided to any new Project Books that have been created with one of the two supported base toolkits. However, depending on which toolkit was chosen there are differences in what happens when a script is started:

Agent Scripting Toolkit – When a script is started, a script ID is automatically created. The compiler generates a call to the Action G_Script_Start to generate the script ID before calling the first Page. The first Page in the script is then launched.

Genesys Agent Interaction Toolkit – This toolkit contains the additional configuration field G_Script_Query_Relationship. This field is a check box (values true, false) that can force at the script startup the automatic navigation to the Process Flow/Stream/Page Genesys/Genesys_Scripts/G_System_Scripts. Along with the ability to supply the script ID and relationship ID values on input to the web application, the following scenarios can occur as shown in Table 74:

Table 74: Scenarios

G_Script_ID	G_Script_RelationshipID	G_Script_Query_Relationship	Action
Yes			Resume the instance of the script identified in the script ID.
No	No		Generate the script ID. Navigate to the first Page.

Table 74: Scenarios (Continued)

G_Script_ID	G_Script_RelationshipID	G_Script_Query _Relationship	Action
No	Yes	False	Generate the script ID. Navigate to the first Page.
No	Yes	True	Navigate to the ProcessFlow/Stream/Page `Genesys/Genesys_Scripts/G_System_Scripts`.

The Agent Scripting compiler automatically generates a branch to the Page `G_System_StartPage`. This Page executes the Action `G_System_Start` which executes the above scenarios.

Agent Features

Genesys Agent Scripting provides a set of pre-defined Fields, Actions, and Pages that allows a script designer to create scripts with the ability to query for agent information and display the information on a Page. These pre-defined elements are automatically copied into a new Project Book when the Project Book is configured with the Genesys Agent Interaction Toolkit.

When the generated Web application is launched, it executes the query for the agent information before displaying the first Page in the script. This results in all agent Fields being populated with their values. The Genesys Agent Interaction Toolkit also provides the Page `G_System_Agent` which displays all agent Fields. The web designer can include this Page in designing a web application, or the web designer can make a copy of the Page and modify it.

Agent Fields

The following lists the Fields related to the Agent that are displayed on the Page `G_System_Agent`:

`G_Agent_ID`. Agent ID. Provided as an input parameter on the URL.

`G_Agent_FirstName`. First name of the agent.

`G_Agent_LastName`. Last name of the agent.

`G_Agent_EmployeeID`. Employee ID of the agent.

`G_Agent_UserName`. User name of the agent.

`G_Agent_Skill_Table`. Table containing the skills of the agent.

`G_Agent_SkillName`. Skill Name. This field is a column in the table `G_Agent_Skill_Table`.

G_Agent_SkillLevel. Skill Level. This field is a column in the table G_Agent_Skill_Table.

G_Agent_AvailableMedia_Table. Table containing the available media of the agent.

G_Agent_AvailableMedia. Available Media. This field is a column in the table G_Agent_AvailableMedia_Table.

G_Agent_DN_Table. Table containing the list of DNs of the agent.

G_Agent_DN_Name. Name of the DN. This field is a column in the table G_Agent_DN_Table.

G_Agent_DN_Status. Status of the DN. This field is a column in the table G_Agent_DN_Table.

G_Agent_DN_LoggedQueue_Table. Table containing logged queues for a specific DN. This table is a column in the table G_Agent_DN_Table.

G_Agent_DN_LoggedQueue. The name of the logged queue. This field is a column in the table G_Agent_DN_LoggedQueue_Table.

Agent Actions

The following Action is related to the agent:

G_Agent_Get. Queries for the agent information

Input Parameters:

G_Agent_ID

G_Config_Place - Provided as an input parameter in the URL

Output Parameters:

G_Agent_FirstName

G_Agent_LastName

G_Agent_EmployeeID

G_Agent_UserName

G_Agent_Skill_Table

G_Agent_AvailableMedia_Table

G_Agent_DN_Table

G_Agent_DN_LoggedQueue_Table

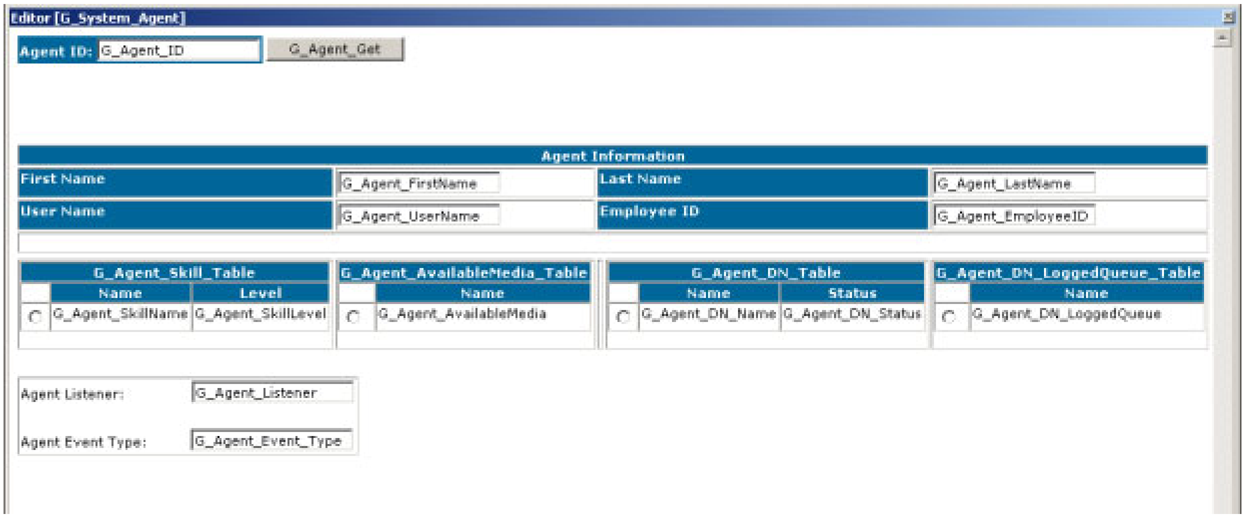
Agent Pages

The following Page is related to the Agent:

Genesys/Genesys_Information/G_System_Agent. Displays the agent information (Fields) along with their Actions.

Note: A script designer can include this Page in the script or create a new agent Page to display the agent information.

Figure 168 shows an example of the `G_System_Agent` Page.



Editor [G_System_Agent]

Agent ID: G_Agent_ID

Agent Information			
First Name	G_Agent_FirstName	Last Name	G_Agent_LastName
User Name	G_Agent_UserName	Employee ID	G_Agent_EmployeeID

G_Agent_Skill_Table		
	Name	Level
○	G_Agent_SkillName	G_Agent_SkillLevel

G_Agent_AvailableMedia_Table	
	Name
○	G_Agent_AvailableMedia

G_Agent_DN_Table		
	Name	Status
○	G_Agent_DN_Name	G_Agent_DN_Status

G_Agent_DN_LoggedQueue_Table	
	Name
○	G_Agent_DN_LoggedQueue

Agent Listener: G_Agent_Listener

Agent Event Type: G_Agent_Event_Type

Figure 168:Example of G_System_Agent Page

Interaction Features

Genesys Agent Scripting provides a set of pre-defined Fields, Actions, and Pages that allows a script designer to create scripts with the ability to perform actions on a specific interaction. These pre-defined elements are automatically copied into a new Project Book when the Project Book is configured with the Genesys Agent Interaction Toolkit.

When the generated Web application is launched and an Interaction ID is provided, the application executes the query for the interaction information before displaying the first Page in the script. This results in all interaction Fields being populated with their values.

Interaction Fields

The following lists the Fields related to the interaction:

`G_AIL_ID`. The Agent Interaction Layer ID. This value is derived from the input parameter Field `G_Interaction_ID`, which may contain the Interaction ID from the T-Server.

`G_Interaction_KVPairTable`. Table containing a list of KVPairs associated with the interaction (attached data).

`G_Interaction_KVPairKey`. KVPair Key. This Field is a column in the table `G_Interaction_KVPairTable`.

G_Interaction_KVPairValueTypes. KVPair value type. This is a dropdown list Field containing the valid value types. This Field is a column in the table G_Interaction_KVPairTable.

G_Interaction_KVPairValue. KVPair Value. This Field is a column in the table G_Interaction_KVPairTable.

G_Interaction_ToNumber. The destination number to transfer the call.

G_Interaction_Location. The location of the destination number to transfer the call.

G_Interaction_TransferMode. A check box instructing the transfer to perform either a one-step or two-step (initiate/complete/retrieve) transfer.

G_Interaction_Attach_Script_Data. A check box instructing the transfer to attach the script data on the transfer. If true, the Script ID value is added as attached data to the interaction with a key value of GAS_ScriptID.

Interaction Actions - Attached Data

The following are the interaction Actions that operate on the interaction's attached data:

G_Interaction_Get_KVPair. Searches a specific interaction for a specific key and returns the value and its type for the key.

Input Parameters:

G_AIL_ID

G_Config_Place

G_Interaction_KVPairKey

Output Parameters:

G_Interaction_KVPairValueTypes

G_Interaction_KVPairValue

G_Interaction_Set_KVPair. Searches a specific interaction for a specific key. If the key is found, its value and type are updated; otherwise, the key along with its value and type are added. The Field G_Interaction_KVPairTable is updated with the current interaction attached data.

Input Parameters:

G_AIL_ID

G_Config_Place

G_Interaction_KVPairKey

G_Interaction_KVPairValueTypes

G_Interaction_KVPairValue

Output Parameter:

G_Interaction_KVPairTable

G_Interaction_Delete_KVPair. Deletes from a specific interaction the key specified in G_Interaction_KVPairKey.

Input Parameters:

G_AIL_ID

G_Config_Place

G_Interaction_KVPairKey

Output Parameter:

G_Interaction_KVPairTable - Entry is removed.

G_Interaction_DeleteSelectedKVPairs. Deletes from a specific interaction all selected entries from the Field table G_Interaction_KVPairTable

Input Parameters:

G_AIL_ID

G_Config_Place

G_Interaction_KVPairTable – Selected entries

Output Parameter:

G_Interaction_KVPairTable – All selected entries are removed.

G_Interaction_DeleteAllKVPairs. Deletes from a specific interaction all entries from the Field table G_Interaction_KVPairTable

Input Parameters:

G_AIL_ID

G_Config_Place

G_Interaction_KVPairTable – All entries

Output Parameter:

G_Interaction_KVPairTable – All entries are removed.

G_Interaction_Refresh. Retrieves attached data for a specific interaction

Input Parameters:

G_AIL_ID

G_Config_Place

Output Parameter:

G_Interaction_KVPairTable – Table is updated with KVPairs from the interaction’s attached data.

Interaction Actions - Media

The following are the interaction Actions that operate on the interaction’s media:

G_Interaction_Disconnect. Disconnects a specific interaction.

Input Parameters:

G_AIL_ID

G_Config_Place

Output Parameter:

None.

G_Interaction_MarkDone. Marks a specific interaction as done.

Input Parameters:

G_AIL_ID

G_Config_Place

Output Parameter:

None.

G_Interaction_InitiateTransfer. Initiates a transfer to another party. This can be either a one-step or the first step of a two-step transfer.

Input Parameters:

G_AIL_ID

G_Config_Place

G_Interaction_ToNumber

G_Interaction_Location

G_Interaction_TransferMode

G_Script_ID

G_Interaction_Attach_Script_Data

Output Parameter:

None.

G_Interaction_CompleteTransfer. Completes a transfer to another party. It is the second step of a two-step transfer and is enabled (made visible) if the Field G_Interaction_TransferMode is set to Two-Step Transfer.

Input Parameters:

G_AIL_ID

G_Config_Place

Output Parameter:

None.

G_Interaction_Retrieve. Retrieves the call that was placed on hold after the first step of a two-step transfer. It is enabled (made visible) if the Field G_Interaction_TransferMode is set to Two-Step Transfer.

Input Parameters:

G_AIL_ID

G_Config_Place

Output Parameter:

None.

G_Interaction_Transfer. Saves the current instance of the script to a database (see "Suspend" on page 354) and then redirects to the Process Flow/Stream/Page Genesys/Genesys_Actions/G_Interaction_Transfer. This action takes no parameters.

Interaction Pages

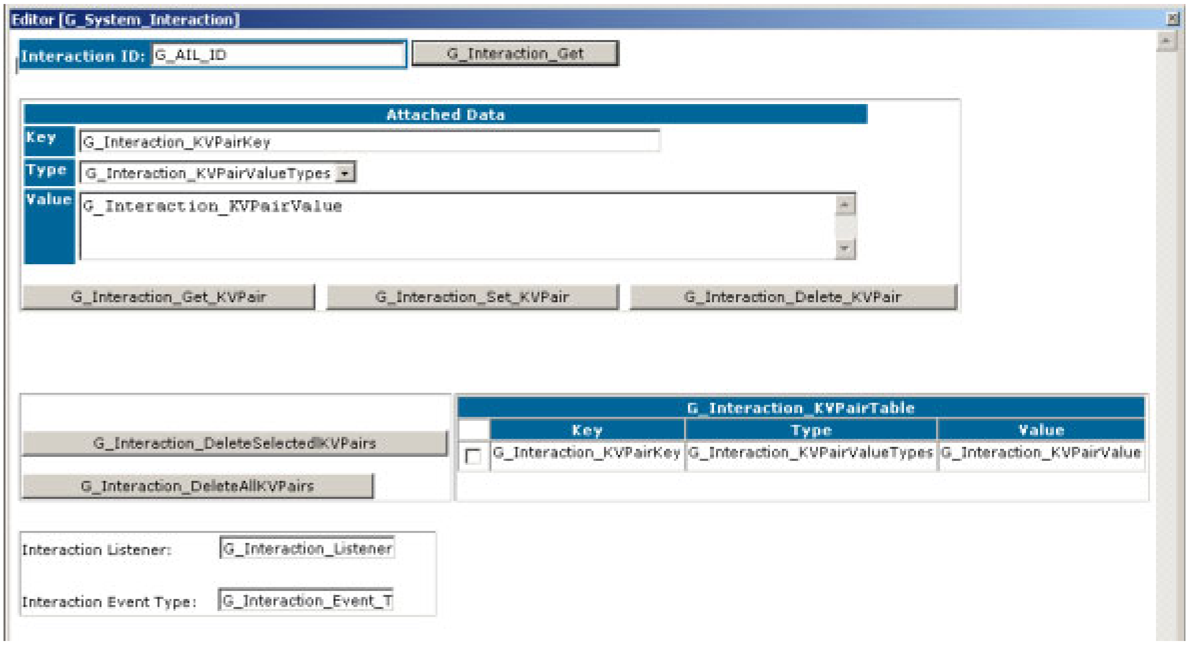
The following Pages are related to the interaction. You can include these Process Flow/Stream/Pages in your scripts or customize them for your environment.

`Genesys/Genesys_Information/G_System_Interaction`. Displays the interaction attached data along with the associated actions. Figure 169 shows an example of the `G_System_Interaction` Page.

Note: A script designer can include this Page in the script or create a new interaction Page to display the attached data.

`Genesys/Genesys_Information/G_Interaction_Transfer`. Displays the information to transfer a call. This Page is displayed when the Action `G_Interaction_Transfer` is executed. Figure 170 shows an example of the `G_Interaction_Transfer` Page.

Note: A script designer can include this Page in the script or create a new transfer Page.


Editor [G_System_Interaction]
G_Interaction_Get
Attached Data
G_Interaction_Get_KVPair
G_Interaction_Set_KVPair
G_Interaction_Delete_KVPair
G_Interaction_DeleteSelectedKVPairs
G_Interaction_DeleteAllKVPairs

Interaction ID: G_AIL_ID

Key	G_Interaction_KVPairKey
Type	G_Interaction_KVPairValueTypes
Value	G_Interaction_KVPairValue

G_Interaction_KVPairTable

	Key	Type	Value
□	G_Interaction_KVPairKey	G_Interaction_KVPairValueTypes	G_Interaction_KVPairValue

Interaction Listener: G_Interaction_Listener

Interaction Event Type: G_Interaction_Event_T

Figure 169:Example of G_System_Interaction Page

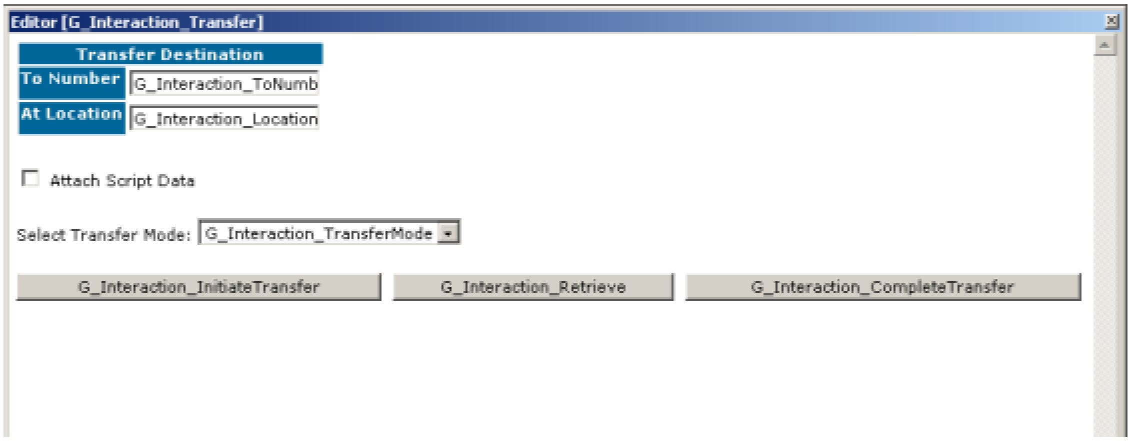


Figure 170:Example of G_Interaction_Transfer Page

Contact Features

Genesys Agent Scripting provides a set of pre-defined Fields, Actions, and Pages that allows a script designer to create scripts with the ability to perform actions on a contact. These pre-defined elements are automatically copied into a new Project Book when the Project Book is configured with the Genesys Agent Interaction Toolkit.

When the generated Web application is launched and a Contact ID is provided, the application executes the query for the contact information before displaying the first Page in the script. This results in all contact Fields being populated with their values.

Contact Fields

The following lists the Fields related to the contact:

`G_Contact_ID`. Contact ID value from the Contact Server. Provided as an input parameter in the URL.

`G_Contact_Title`. Title. (For example, `Mr`.)

`G_Contact_FirstName`. First name.

`G_Contact_LastName`. Last name.

`G_Contact_Email_Table`. Table of e-mail addresses.

`G_Contact_Email_ID`. Internal ID of the e-mail address. This is a column in the table `G_Contact_Email_Table`. However, the column is not displayed. The ID is used to execute an Action on a specific e-mail address.

`G_Contact_Email_Address`. e-mail address. This is a column in the table `G_Contact_Email_Table`.

`G_Contact_Email_Description`. e-mail description. This is a column in the table `G_Contact_Email_Table`.

`G_Contact_Email_IsPrimary`. Indicates whether the e-mail address is primary. This is a column in the table `G_Contact_Email_Table`.

`G_Contact_Phone_Table`. Table of phone numbers.

`G_Contact_Phone_ID`. Internal ID of the phone number. This is a column in the table `G_Contact_Phone_Table`. However, the column is not displayed. The ID is used to execute an Action on a specific phone number.

`G_Contact_Phone_Number`. Phone number. This is a column in the table `G_Contact_Phone_Table`.

`G_Contact_Phone_Description`. Phone description. This is a column in the table `G_Contact_Phone_Table`.

`G_Contact_Phone_IsPrimary`. Indicates whether the phone number is primary. This is a column in the table `G_Contact_Phone_Table`.

Contact Actions

The following lists the Actions related to the contact:

`G_Contact_Get`. Retrieves the information for a specific contact from the Contact Server.

Input Parameter:

```
G_Contact_ID
```

Output Parameters:

```
G_Contact_Title
G_Contact_FirstName
G_Contact_LastName
G_Contact_Email_Table
G_Contact_Phone_Table
```

`G_Contact_Create`. Creates a new contact in the Contact Server.

Input Parameters:

```
G_Contact_Title
G_Contact_FirstName
G_Contact_LastName
G_Contact_Email_Table
G_Contact_Phone_Table
```

Output Parameter:

```
G_Contact_ID
```

`G_Contact_Delete`. Deletes a specific contact from the Contact Server.

Input Parameter:

```
G_Contact_ID
```

Output Parameter:

None.

`G_Contact_Update`. Updates the Title, First and Last Name for a specific contact in the Contact Server.

Input Parameters:

`G_Contact_ID`

`G_Contact_Title`

`G_Contact_FirstName`

`G_Contact_LastName`

Output Parameter:

None.

`G_Contact_Clear`. Clears all contact Fields.

Input Parameter:

None.

Output Parameters:

`G_Contact_Title`

`G_Contact_FirstName`

`G_Contact_LastName`

`G_Contact_Email_Table`

`G_Contact_Phone_Table`

`G_Contact_Add_Email`. Adds a new e-mail address for a specific contact in the Contact Server. Duplicate e-mail addresses are not allowed. Also, adding a new primary e-mail address will make all other primary e-mail addresses secondary. Only one e-mail address can be primary.

Input Parameters:

`G_Contact_ID`

`G_Contact_Email_Address`

`G_Contact_Email_IsPrimary`

`G_Contact_Email_Description`

Output Parameter:

`G_Contact_Email_Table` – New e-mail address is added.

`G_Contact_Update_Email`. Updates an existing e-mail address for a specific contact in the Contact Server. Duplicate e-mail addresses are not allowed. Also, updating an e-mail address to primary will make all other primary e-mail addresses secondary. Only one e-mail address can be primary.

Input Parameters:

`G_Contact_ID`

`G_Contact_Email_ID`

`G_Contact_Email_Address`

`G_Contact_Email_IsPrimary`

`G_Contact_Email_Description`

Output Parameter:

`G_Contact_Email_Table`—e-mail address is updated.

`G_Contact_Delete_Email`. Deletes an existing e-mail address for a specific contact in the Contact Server.

Input Parameters:

`G_Contact_ID`

`G_Contact_Email_ID`

Output Parameter:

`G_Contact_Email_Table`—e-mail address is deleted.

`G_Contact_Add_Phone`. Adds a new phone number for a specific contact in the Contact Server. Duplicate phone numbers are not allowed. Also, adding a new primary phone number will make all other primary phone numbers secondary. Only one phone number can be primary.

Input Parameters:

`G_Contact_ID`

`G_Contact_Phone_Number`

`G_Contact_Phone_IsPrimary`

`G_Contact_Phone_Description`

Output Parameter:

`G_Contact_Phone_Table`—New phone number is added.

`G_Contact_Update_Phone`. Updates an existing phone number for a specific contact in the Contact Server. Duplicate phone numbers are not allowed. Also, updating an phone number to primary will make all other primary phone numbers secondary. Only one phone number can be primary.

Input Parameters:

`G_Contact_ID`

`G_Contact_Phone_ID`

`G_Contact_Phone_Number`

`G_Contact_Phone_IsPrimary`

`G_Contact_Phone_Description`

Output Parameter:

`G_Contact_Phone_Table`—Phone number is updated.

`G_Contact_Delete_Phone`. Deletes an existing phone number for a specific contact in the Contact Server.

Input Parameters:

`G_Contact_ID`

`G_Contact_Phone_ID`

Output Parameter:

`G_Contact_Phone_Table`—Phone number is deleted.

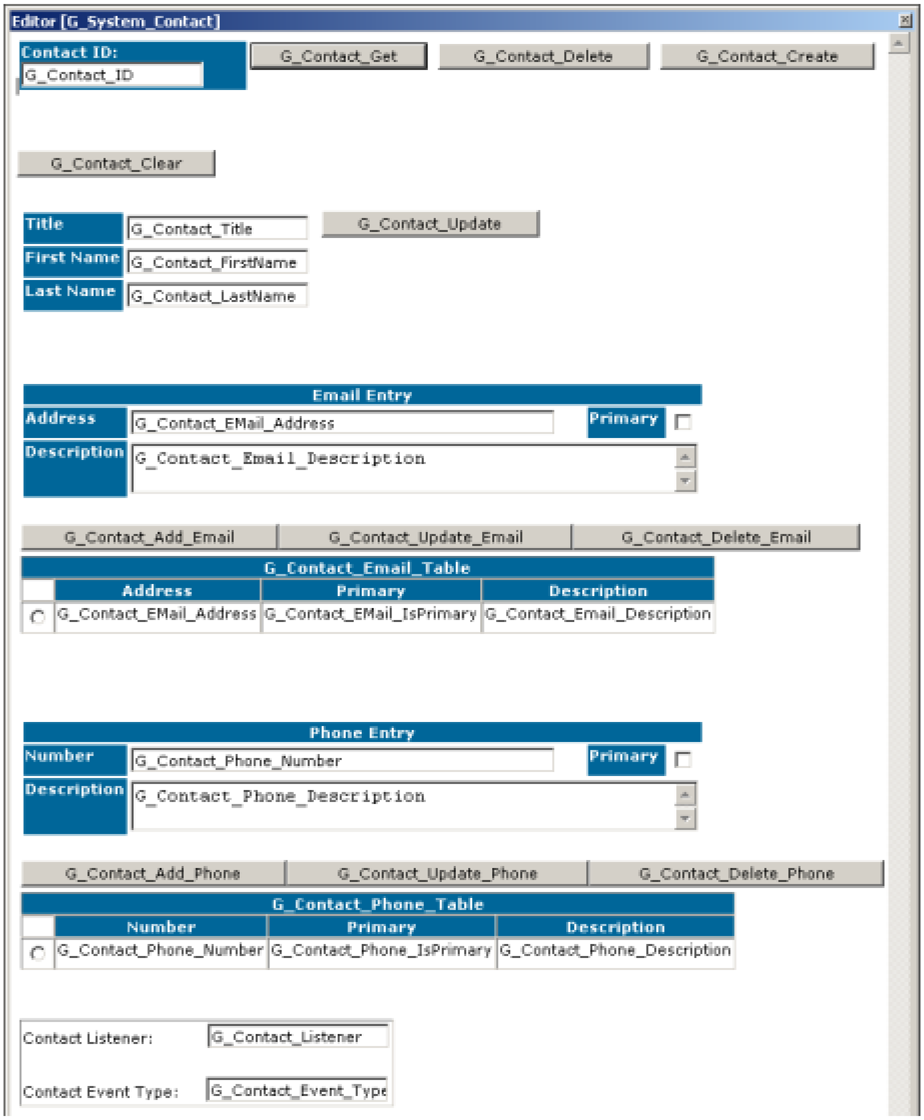
Contact Pages

The following Page is related to the contact. You can include this Process Flow/Stream/Page in your scripts or customize it for your environment.

`Genesys/Genesys_Information/G_System_Contact`. Displays the contact information (Fields) along with the associated Actions.

Note: A script designer can include this page in the script or create a new contact Page to display the contact information.

Figure 171 shows an example of the `G_System_Contact` Page.



Editor [G_System_Contact]

Contact ID: G_Contact_ID

Title: G_Contact_Title

First Name: G_Contact_FirstName

Last Name: G_Contact_LastName

Email Entry

Address: G_Contact_EMail_Address Primary □

Description: G_Contact_Email_Description

G_Contact_Email_Table

	Address	Primary	Description
○	G_Contact_EMail_Address	G_Contact_EMail_IsPrimary	G_Contact_Email_Description

Phone Entry

Number: G_Contact_Phone_Number Primary □

Description: G_Contact_Phone_Description

G_Contact_Phone_Table

	Number	Primary	Description
○	G_Contact_Phone_Number	G_Contact_Phone_IsPrimary	G_Contact_Phone_Description

Contact Listener: G_Contact_Listener

Contact Event Type: G_Contact_Event_Type

Figure 171:Example of G_System_Contact Page

Outbound Features

Genesys Agent Scripting provides a set of pre-defined Fields, Actions, and Pages that allows a script designer to create scripts with the ability to perform actions on an outbound record. These pre-defined elements are automatically copied into a new Project Book when the Project Book is configured with the Genesys Agent Interaction Toolkit.

When the generated Web application is launched and an outbound record handle ID is provided, the application executes the query for the outbound record information before displaying the first Page in the script. This results in all outbound Fields being populated with their values.

Outbound Fields

The following lists the Fields related to outbound:

G_AIL_ID. The Agent Interaction Layer ID. This value is derived from the input parameter Field G_OR_ID which contains the active outbound record handle ID.

G_OR_AddPhone. A String Field to specify a new outbound phone number. This is an input Field to specify the phone number when adding a new outbound record.

G_OR_DoNotCall_Message. A String Field for a Do Not Call message. This is an input Field to specify reason text when issuing the outbound request Do Not Call.

G_OR_CallbackType. A Dropdown List Field to specify the callback rescheduled type (CAMPAIGN or PERSONAL).

G_OR_Reschedule_Time. A Date and Time Field to specify the callback rescheduled time.

G_OR_Reschedule_Date. A Calendar Field to specify a callback rescheduled date.

G_OR_Table. A Table Field containing the list of outbound records. This includes the active record and optionally any chained records.

G_OR_Handle_ID. A String Field (column) in the Table Field G_OR_Table to represent the handle ID for each outbound record (active and chained) in the table.

G_OR_Status. A Dropdown List (column) Field in the Table Field G_OR_Table to represent the status for each outbound record (active and chained) in the table.

G_OR_Active. A Check Box Field (column) in the Table Field G_OR_Table to indicate which outbound record in the table is active. Only one outbound record can be active.

G_OR_Phone. A String Field (column) in the Table Field G_OR_Table to represent the callback phone number for each outbound record (active and chained) in the table.

G_OR_CallResult. A Dropdown List Field (column) in the Table Field G_OR_Table to represent the call result for each outbound record (active and chained) in the table.

G_OR_Campaign_Name. A String Field (column) in the Table Field G_OR_Table to represent the campaign name for each outbound record (active and chained) in the table.

G_OR_Campaign_Mode. A Dropdown List Field (column) in the Table Field G_OR_Table to represent the campaign mode for each outbound record (active and chained) in the table.

G_OR_Campaign_Status. A String Field (column) in the Table Field G_OR_Table to represent the campaign status for each outbound record (active and chained) in the table.

G_OR_Campaign_Description. A String Field (column) in the Table Field G_OR_Table to represent the campaign description for each outbound record (active and chained) in the table.

G_OR_CallListName. A String Field (column) in the Table Field G_OR_Table to represent the CallList name for each outbound record (active and chained) in the table.

G_OR_CF_Table. A Table Field (column) in the Table Field G_OR_Table to represent custom fields for each outbound record (active and chained) in the table.

G_OR_CF_Key. A String Field (column) in the Table Field G_OR_CF_Table to represent the key for each custom field.

G_OR_CF_Type. A Dropdown List Field (column) in the Table Field G_OR_CF_Table to represent the type for each custom field.

G_OR_CF_Value. A String Field (column) in the Table Field G_OR_CF_Table to represent the value for each custom field.

G_OR_CF_DateTime_Value. A Date and Time Field (column) in the Table Field G_OR_CF_Table to represent the Date/Time value for those custom fields of type DATETIME.

G_OR_CF_Integer_Value. A Numeric-Integer Field (column) in the Table Field G_OR_CF_Table to represent the integer value for those custom fields of type INTEGER.

G_OR_CF_Float_Value. A Numeric-Floating Decimal Point Field (column) in the Table Field G_OR_CF_Table to represent the floating point value for those custom fields of type FLOAT.

G_OR_CF_String_Value. A String Field (column) in the Table Field G_OR_CF_Table to represent the string value for those custom fields of type STRING.

Outbound Actions

The following lists the Actions related to outbound:

`G_Outbound_Reject`. Requests the Outbound Campaign Server to reject the callback.

Input Parameter:

`G_AIL_ID`

Output Parameter:

None.

`G_Outbound_Cancel`. Requests the Outbound Campaign Server to cancel the callback.

Input Parameter:

`G_AIL_ID`

Output Parameter:

None.

`G_Outbound_Add_Record`. Requests the Outbound Campaign Server to add a new outbound record for a specific phone number.

Input Parameter:

`G_AIL_ID`

G_OR_AddPhone

Output Parameter:

None.

`G_Outbound_DoNotCall`. Requests the Outbound Campaign Server to mark the callback as `Do Not Call`.

Input Parameter:

`G_AIL_ID`

`G_OR_DoNotCall_Message`

Output Parameter:

None.

`G_Outbound_Reschedule`. Requests the Outbound Campaign Server to reschedule the callback for a later date/time.

Input Parameters:

`G_AIL_ID`

`G_OR_CallbackType`

`G_OR_Reschedule_Time`

`G_OR_Reschedule_Date`

Output Parameter:

None.

G_Outbound_Get_Active_Record. Requests the Outbound Campaign Server to return the active record.

Input Parameter:

G_AIL_ID

Output Parameter:

G_OR_Table – Updated with just the active record.

G_Outbound_Get_All_Record. Requests the Outbound Campaign Server to return all outbound records. This includes the active and any chained records.

Input Parameter:

G_AIL_ID

Output Parameter:

G_OR_Table – Updated with all records.

G_Outbound_Set_Record_Active. Requests the Outbound Campaign Server to make active a specific outbound record.

Input Parameters:

G_AIL_ID

G_OR_Handle_ID

Output Parameter:

G_OR_Table – Updated with the new active record.

G_Outbound_Update_Record. Requests the Outbound Campaign Server to update a specific outbound record.

Input Parameters:

G_AIL_ID

G_OR_Handle_ID

G_OR_Status

G_OR_Phone

G_OR_CallResult

G_OR_CF

Output Parameter:

None.

G_Outbound_Update_CF. Updates the Table Field G_OR_CF with the new value for the current row. This does not update the outbound record custom fields in the Outbound Contact Server. To update the Outbound Contact Server, a G_Outbound_Update_Record must be issued.

Input Parameters:

G_OR_CF_Type

G_OR_CF_String_Value

G_OR_CF_DateTime_Value

G_OR_CF_Integer_Value

G_OR_CF_Float_Value

Output Parameter:

G_OR_CF_Table—Updated with the changes to the custom field.

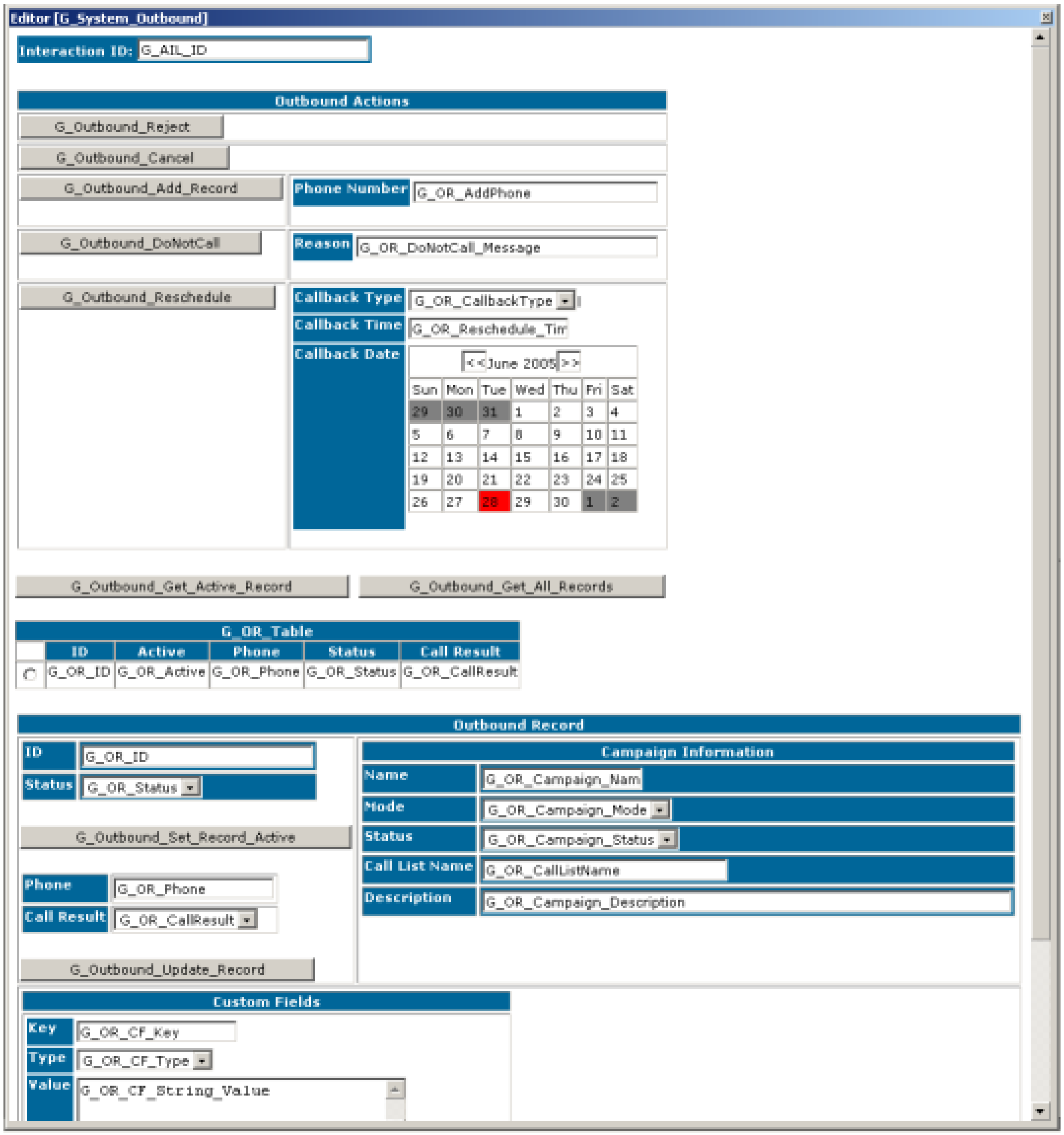
Outbound Pages

The following Page is related to outbound. You can include this Process Flow/Stream/Page in your scripts or customize it for your environment.

Genesys/Genesys_Information/G_System_Outbound. Displays the outbound information (Fields) along with the associated Actions.

A script designer can include this Page in the script or create a new outbound Page to display the outbound information.

Figure 172 shows an example of the G_System_Outbound Page.



Interaction ID: G_AIL_ID

Phone Number: G_OR_AddPhone

Reason: G_OR_DoNotCall_Message

Callback Type: G_OR_CallbackType

Callback Time: G_OR_Reschedule_Tim

Callback Date: << June 2005 >>

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2

G_OR_Table

	ID	Active	Phone	Status	Call Result
○	G_OR_ID	G_OR_Active	G_OR_Phone	G_OR_Status	G_OR_CallResult

ID: G_OR_ID

Status: G_OR_Status

Phone: G_OR_Phone

Call Result: G_OR_CallResult

Name: G_OR_Campaign_Nam

Mode: G_OR_Campaign_Mode

Status: G_OR_Campaign_Status

Call List Name: G_OR_CallListName

Description: G_OR_Campaign_Description

Key: G_OR_CF_Key

Type: G_OR_CF_Type

Value: G_OR_CF_String_Value

Figure 172:Example of G_System_Outbound Page

Reporting

The reporting feature provides the capability to track Process Flows, Pages, and Fields of interest. Information is stored in a database that can be retrieved selectively or formatted by the user as needed.

In order to use Genesys Agent Scripting Reporting, the Base Toolkit of the Project Book must be either the Agent Scripting toolkit or the Genesys Agent Interaction toolkit.

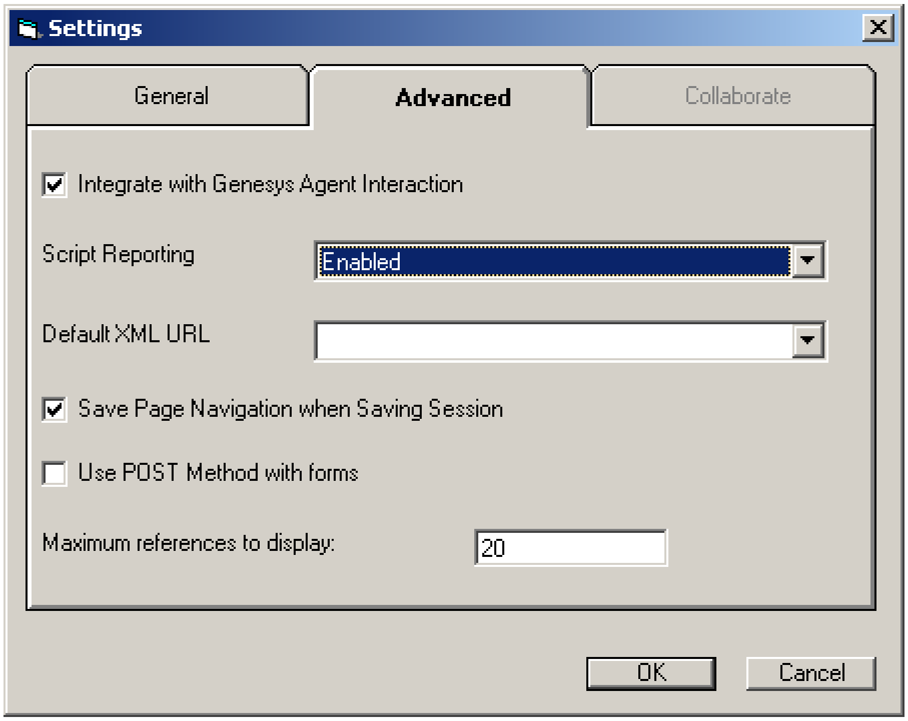
Settings

A few settings must be configured for reporting to function.

Procedure: Enable Standalone Reporting

Start of procedure

1. Select `File` > `Settings`, then select the `Advanced` tab.
2. Select `Enabled` from the `Script Reporting` dropdown list as shown in Figure 173. If this Field is disabled, nothing will be reported.



☑ Integrate with Genesys Agent Interaction

Script Reporting: Enabled

Default XML URL:

☑ Save Page Navigation when Saving Session

□ Use POST Method with forms

Maximum references to display: 20

Figure 173:Script Reporting - Enabled

End of procedure

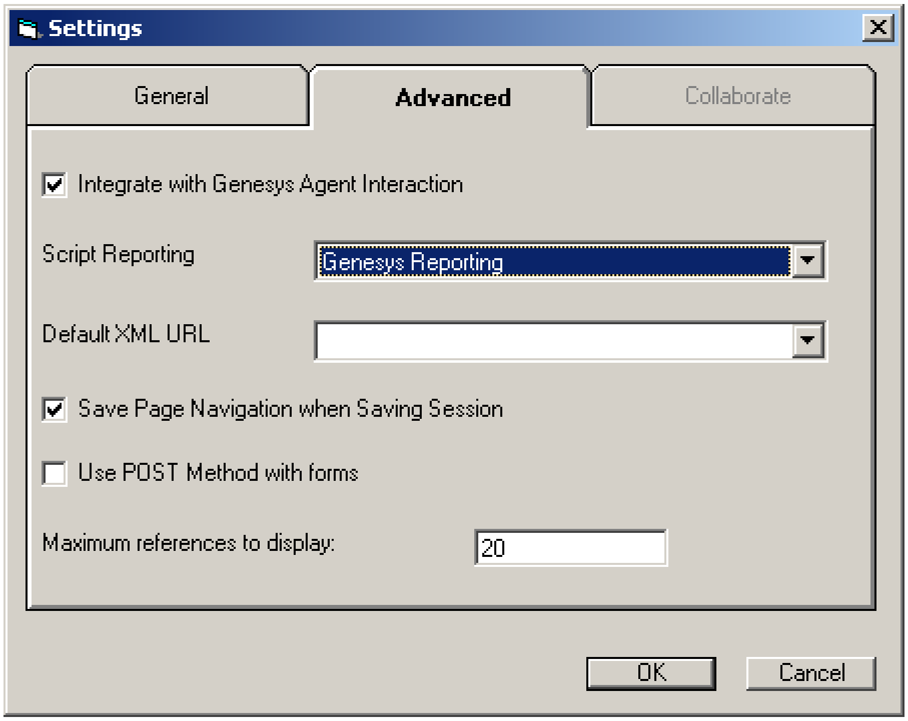
Procedure: Enable Reporting With the Genesys Agent Interaction Layer

Start of procedure

1. Select `File` > `Settings`, then select the `Advanced` tab.
2. Select `Genesys Reporting` from the `Script Reporting` dropdown list as shown in Figure 174.

   The `Integrate with Genesys Agent Interaction` check box must be selected to enable this Script Reporting choice.

Note: This type of Reporting is only enabled for Project Books created with the Genesys Agent Interaction Toolkit.



☑ Integrate with Genesys Agent Interaction

Script Reporting: Genesys Reporting

Default XML URL:

☑ Save Page Navigation when Saving Session

□ Use POST Method with forms

Maximum references to display: 20

Figure 174:Script Reporting - Genesys Reporting

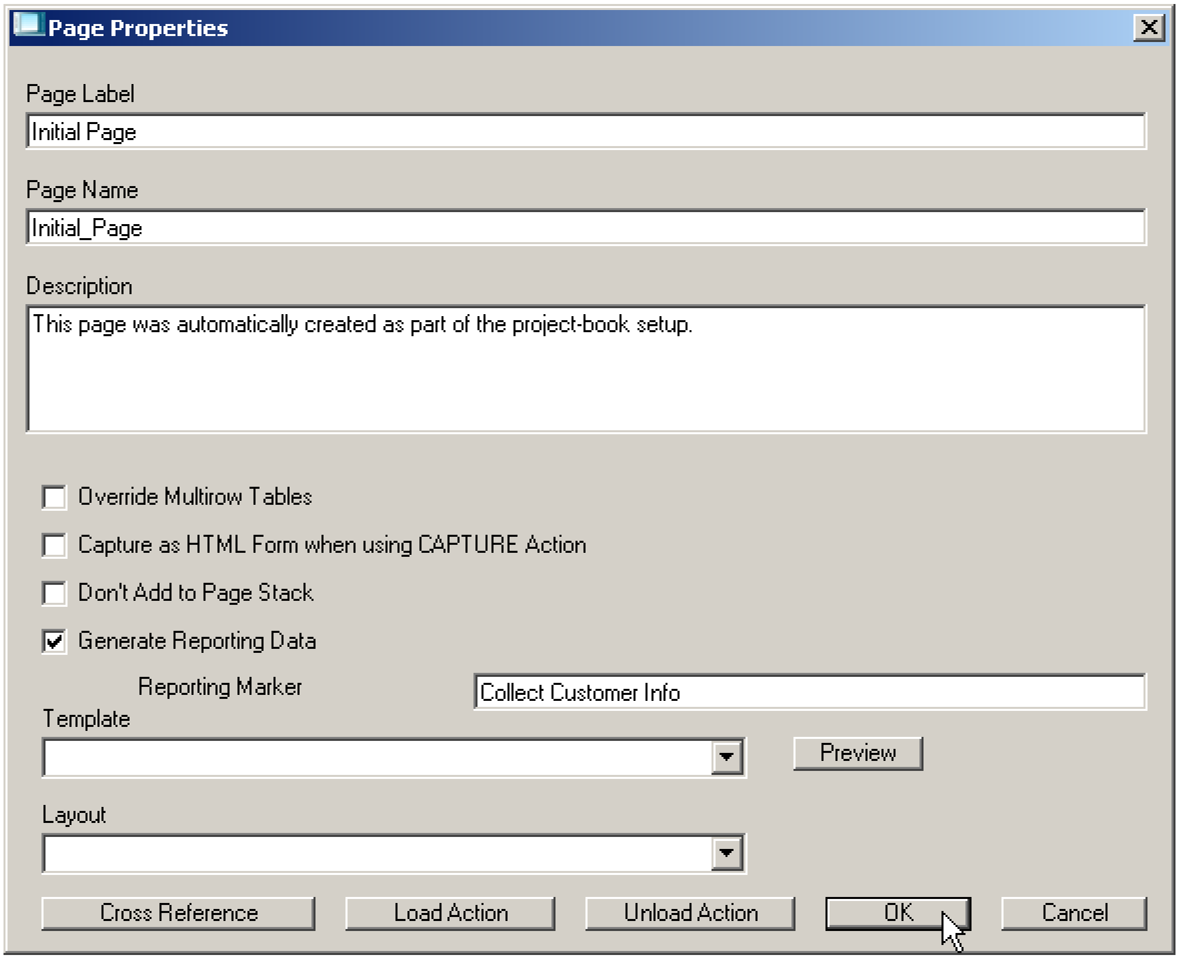
End of procedure

Procedure: Set Reporting for the Page

Start of procedure

1. In the `Page Properties` dialog box of the Page for which reporting activity will be done, make sure to select the `Generate Reporting Data` check box as shown in Figure 175. If this is not selected, even the Fields that are set to be reported will not be reported.
2. Enter some meaningful keywords to search on when reports are generated in the `Reporting Marker` Field.

> **Note:** The Reporting Marker Field is another dimension defined by the script designer, used to organize Page statistics and data. For example, a Process Flow may contain several Pages used to collect customer information. The script designer may want to group these Pages for reporting purposes by entering the string "Collect Customer Info" in the Reporting Marker Field.



Page Label: Initial Page

Page Name: Initial_Page

Description: This page was automatically created as part of the project-book setup.

□ Override Multirow Tables

□ Capture as HTML Form when using CAPTURE Action

□ Don't Add to Page Stack

☑ Generate Reporting Data

Reporting Marker: Collect Customer Info

Template:

Layout:

Figure 175:Page Properties Dialog Box - Generate Reporting Data

End of procedure

Procedure: Set Reporting for the Fields on the Page

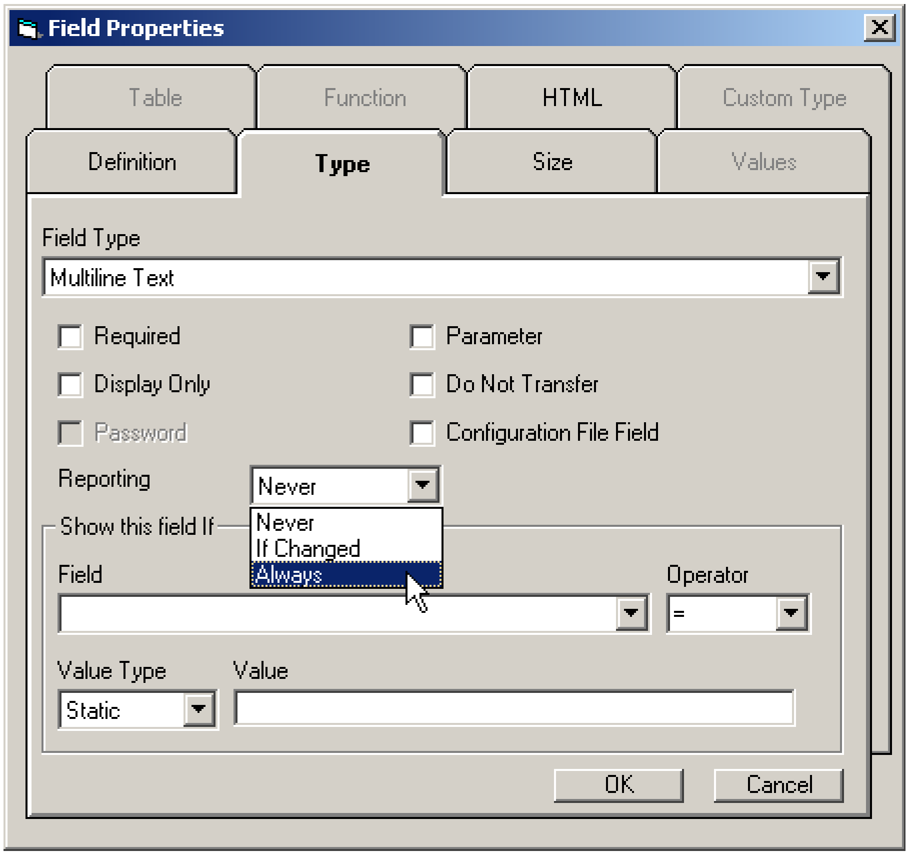
Purpose: To set the Fields of the Page that will be reported accordingly.

Start of procedure

1. Select the Field and open its `Field Properties` dialog box.
2. Select the `Type` tab as shown in Figure 176.
3. The `Reporting` dropdown list has three choices:
   - `Never` - do not report
   - `If Changed` - report only when the value of the Field has changed
   - `Always` - always report this Field

Select your reporting choice for this Field.

4. Repeat the previous steps for other Fields on the Page as appropriate.



Field Type: Multiline Text

□ Required
□ Parameter
□ Display Only
□ Do Not Transfer
□ Password
□ Configuration File Field

Reporting: Never

Show this field If

Field:

Operator: =

Value Type: Static

Value:

Figure 176:Reporting Field in Field Properties Dialog Box

End of procedure

Table Structures

Genesys Agent Scripting has provided SQL scripts to create tables for the supported databases (Oracle, DB2, Microsoft SQL). The corresponding SQL script of the database has to be run only once in the beginning before creating any reporting. The SQL scripts can be found in the folder:

...`\GenesysIntegration\sql`

There are three tables created by the SQL script:

- `GASREPORTINGPROCFLOW`
- `GASREPORTINGPAGE`
- `GASREPORTINGDATA`

GASREPORTINGPROCFLOW

The GASREPORTINGPROCFLOW table is used to report on a Process Flow and has the following fields:

PROCFLOWID. The unique ID of this GASREPORTINGPROCFLOW entry.

SCRIPTID. The unique Script ID for this Process Flow instance.

FLOWNAME. The name of the Process Flow.

VERSION. The version of the Project Book, as specified in the Project Book Properties.

STARTTIME. The starting time of the Process Flow (as in the time the first Page is loaded).

ENDTIME. The ending time of the Process Flow (as in the time the last Page is unloaded). The last Page will be the one in which the PROCESSFLOWRESULT is set to “Completed” or “Abandoned.”

PROCESSFLOWRESULT. The result of the Process Flow: Started, Abandoned, Completed, Suspended, Resumed, Transferred.

GASREPORTINGPAGE

The GASREPORTINGPAGE table is used for reporting on a page and has the following fields:

SCRIPTID. The unique Script ID for this Process Flow instance.

SEQUENCEID. This is the Page order, starting with 0.

PAGENAME. The name of this Page.

STARTTIME. The time that this Page was loaded.

ENDTIME. The time that this Page was unloaded.

PROCESSFLOWRESULT. The result of the Process Flow at the time this Page is unloaded: Started, Abandoned, Completed, Suspended, Resumed, Transferred. This is the value from the Field G_Reporting_Process_Flow_Result.

BUSINESSRESULT. User-specified result of the Business process. This is the value from the Field G_Reporting_Business_Result.

INTERACTIONID. An interaction ID from either T-Server or AIL (Agent Interaction Layer). This will only be set if the Project Book was created with the Genesys Agent Interaction Toolkit.

AGENTID. The ID of an agent. This will only be set if the Project Book was created with the Genesys Agent Interaction Toolkit.

MARKER. This contains the data entered in the Reporting Marker Field on the Page Properties for this Page.

PAGEREPORTID. The unique ID of this entry.

GASREPORTINGDATA

The GASREPORTINGDATA table contains the Fields the user is requesting to be included with reporting data. All Fields will have at least one entry, created when the first Page of the Process Flow Instance is loaded, with the initial value of all Fields. Subsequently, any Field marked as Reporting Always in the Field properties will have an entry created every time Page statistics are gathered. Any Field marked as Reporting If Changed in the Field properties will have an entry created when Page statistics are gathered only if its value has changed since the last time Page statistics have been gathered.

FIELDNAME. The name of the Field.

FIELDTYPE. The type of the Field. Only String and Integer values are supported. The String value for all Field types will be used except for Fields of type Numeric with a format of Integer, which will have their value stored as an Integer.

FIELDINTEGERVALUE. This is the value of the Field as an Integer when the Field type is Integer.

FIELDSTRINGVALUE. This is the string representation of the Field when the Field type is String.

PAGEREPORTID. The ID from the GASREPORTINGPAGE table that corresponds to the Page unload for this Field value collection point.

SCRIPTID. The unique Script ID for this Process Flow instance.

DATAREPORTID. The unique ID of this entry.

Samples

In GenesysToolkit (a Project Book provided with Genesys Agent Scripting) there are samples of Fields and scripting Pages for reporting and for integrating with the Genesys Agent Interaction Layer. For example:

- G_Reporting_Process_Flow_Result and G_Reporting_Business_Result Fields are used for the GASREPORTINGPAGE table.
- G_System_Interaction Page is used for attached data when working with Genesys Agent Desktop.

The Fields and Pages can be imported as necessary, or the toolkit can be specified in the Base Toolkit during Project Book creation.

The WWGReporter Project Book is a sample showing you how to create sample reports and manage the Reporting database. This is a fully-functional Project Book that can be enhanced and modified to meet the end user's needs.

Chapter

15 Integrating with Genesys Software

This chapter provides an overview of how Genesys Agent Scripting integrates with the Genesys Outbound Contact Server component. It also introduces the Assigner Servlet for setting up Assignment Rules.

This chapter contains the following sections:

Outbound Contact Server

This section describes integration of Genesys Agent Scripting with Genesys Outbound Contact Server.

Genesys Agent Scripting can create a Script object in Configuration Server that represents a given Process Flow. When the Process Flow–to–Script Object association has been created in Configuration Server, these Script objects can be associated with Outbound Campaign Groups, Campaigns, and Calling Lists. One of the assignment criteria that is used by the Genesys Agent Scripting Assigner is based upon the Script object's association to a Campaign Group, Campaign, or Calling List.

The Script object names are created by using a combination of the name of the Project Book and the Process Flow. For example, if Project Book is `Insurance` and Process Flow is `Auto`, the Script object in Configuration Server will be named `Insurance_Auto`. All Script objects are created with the type `Outbound Campaign`.

Genesys Agent Scripting communicates with Configuration Server by using the SOAP protocol. By default, Configuration Server does not have the SOAP protocol enabled, so you must update the Configuration Server's configuration file to add in the SOAP section. This procedure is discussed further in the *Genesys Agent Scripting 8.1 Deployment Guide*.

Note: Restarting any web server—GAD or GAS Tomcats, IIS, or Genesys Integration Server—requires that you also restart all Outbound campaigns.

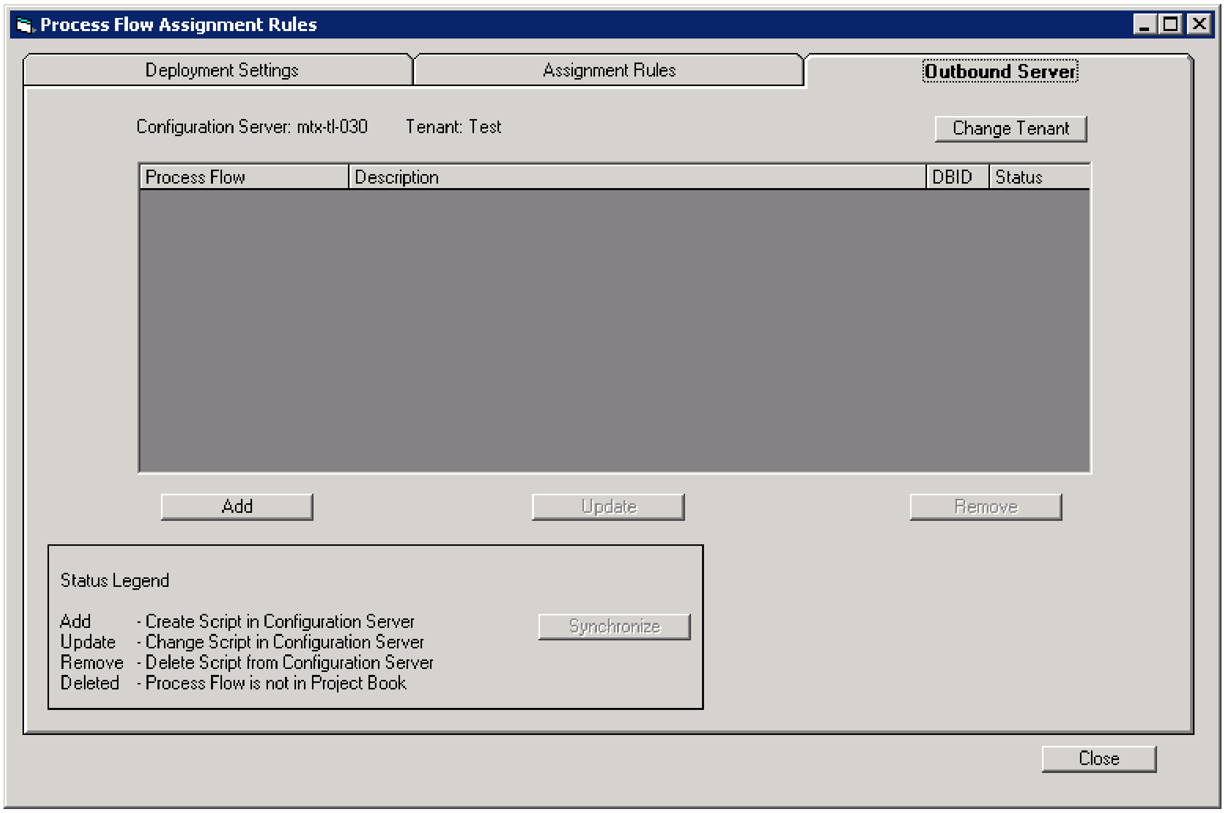
Creating a Script Object–to–Process Flow Association

Procedure: Creating a new Script object

Use the `Compile` menu to perform this procedure.

Start of procedure

1. Select `Compile` >`Assignment Rules` to open the `Process Flow Assignment Rules` dialog box as shown in Figure 177.
2. Select the `Outbound Server` tab.



Configuration Server: mtx-tl-030   Tenant: Test

Process Flow	Description	DBID	Status

Status Legend

Add - Create Script in Configuration Server
Update - Change Script in Configuration Server
Remove - Delete Script from Configuration Server
Deleted - Process Flow is not in Project Book

Figure 177:Process Flow Assignment Rules - Outbound Server Tab

Outbound Server Tab Options

Table 75 describes the options on the Outbound Server tab of the Process Flow Assignment Rules dialog box.

Table 75: Outbound Server Tab Options

Control	Description
Configuration Server label	Name of the current Configuration Server.
Tenant label	Tenant is a label showing the current working tenant where Assignment Rules and Outbound Campaign script objects are stored. An association with a tenant is made after a Change Tenant, Deploy Rules, Remove Rules, or Synchronize operation has been performed. If the association have not been made, the value is absent.
Change Tenant button	Click Change Tenant to set Configuration Server and Tenant where Assignment Rules for Interaction Workspace and Outbound Campaign script objects will be deployed. Clicking this button opens the Configuration Login dialog box, where you enter the user name, password, application, host, and SOAP port to connect to Configuration Server. After a successful connection to Configuration Server, a dialog box that contains the list of valid tenants is displayed. You can select a tenant from the list. The Tenant label is also updated with the name of the tenant that you select.
Add button	Click Add to add a Process Flow and description to the Outbound table.
Update button	Click Update to edit or update the description of a Process Flow that already is in the Outbound table.
Remove button	Click Remove to mark the selected Process Flow for removal from the Outbound table.

Table 75: Outbound Server Tab Options (Continued)

Control	Description
Synchronize button	Click Synchronize to apply all Outbound table changes to the Configuration Server. Clicking this button opens the Configuration Login dialog box where you enter the user name, p, application, host, and SOAP port to connect to Configuration Server. If a tenant is associated with the Outbound table (the tenant name would be displayed in the upper left-hand corner), all operations will be applied to that tenant. Otherwise, after successful connection to the Configuration Server, a dialog box that contains the list of valid tenants will be displayed. You then select a tenant to which all operations will be applied.
Close button	Click Close to close the Process Flow Assignment Rules dialog box.

Outbound Table Synchronize Status Column

The Outbound table contains a status column that determines the action that is to be performed when a Synchronize operation is issued. If the status column is empty, then Configuration Server is up to date with the Outbound table. Otherwise, the status column can contain one of the following four values as shown in Table 76:

Table 76: Status of Synchronize Operation

Value	Meaning
Add	Creates a Script object in Configuration Server. The DBID of the script object will be propagated back to the Outbound table, if no error is encountered.
Update	Updates the Script object in Configuration Server with the changes that have been made in the Outbound table.
Remove	Deletes the Script object from Configuration Server and remove the entry from the Outbound table.
Deleted	The Script object in Configuration Server is associated with a Process Flow that is no longer in this Project Book. To remove this entry from the Outbound table, a Remove operation should be issued against that entry, followed by a Synchronize operation.

3. Click Add to add an entry to the Outbound table.

4. Select a Process Flow from the dropdown list as shown in Figure 178.

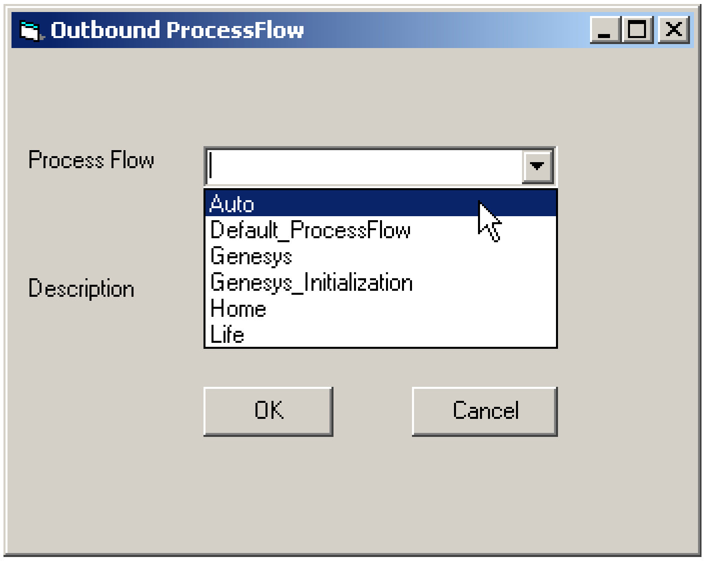
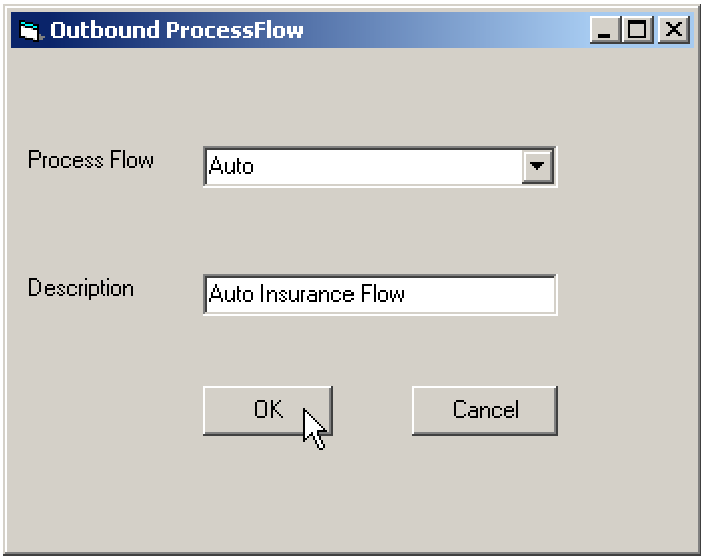


Figure 178:Outbound Process Flow Dialog Box

5. Enter a description into the `Description` field, and then click `OK`, as shown in Figure 179.

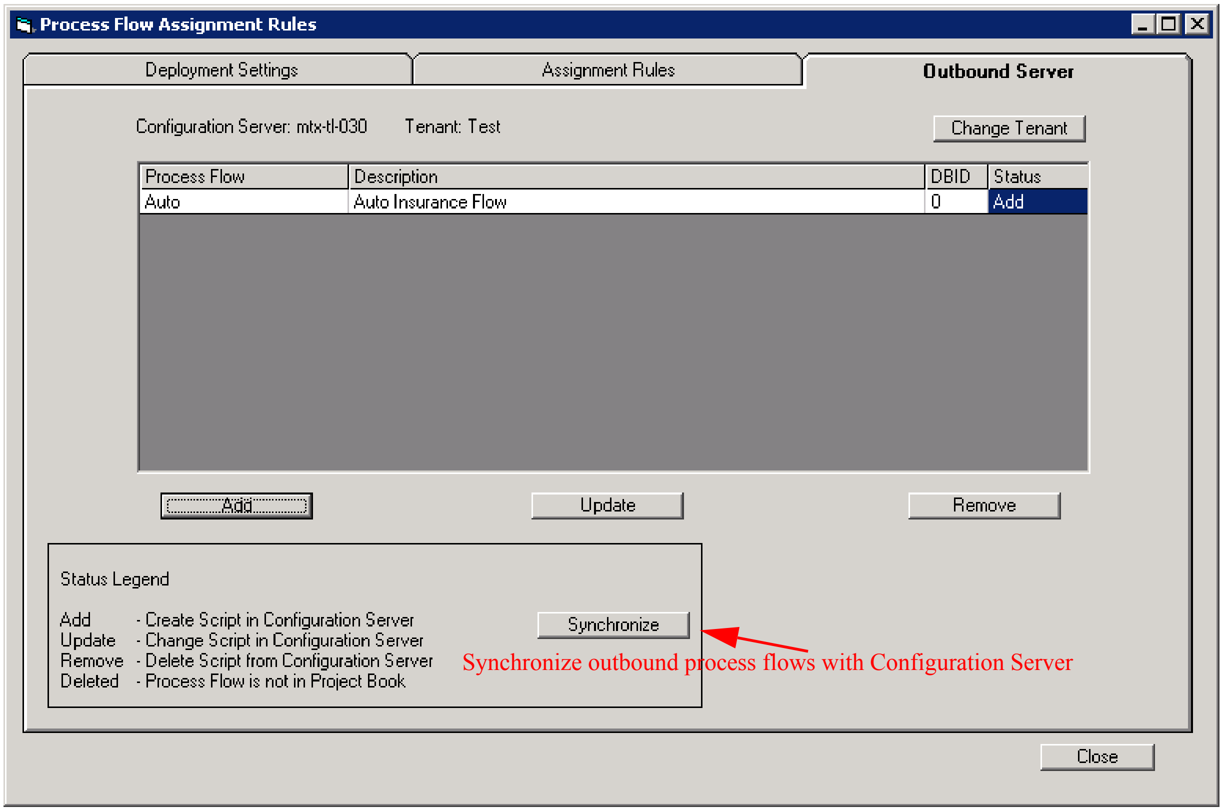


Process Flow: Auto

Description: Auto Insurance Flow

Figure 179:Outbound Process Flow Dialog Box - Description

The entry now appears in the Outbound table as shown in Figure 180. Note that the `DBID` column shows `0` and that the status is `Add`. The changes have not been saved to Configuration Server at this time.



Configuration Server: mtx-tl-030 Tenant: Test

Process Flow	Description	DBID	Status
Auto	Auto Insurance Flow	0	Add

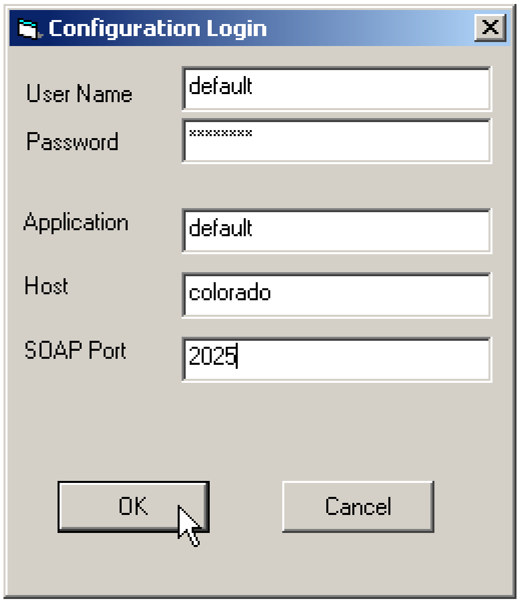
Status Legend

Add - Create Script in Configuration Server
Update - Change Script in Configuration Server
Remove - Delete Script from Configuration Server
Deleted - Process Flow is not in Project Book

Synchronize outbound process flows with Configuration Server

Figure 180:Outbound Server Tab - Process Flow Added

6. Click `Synchronize` to create a Script object in Configuration Server that is associated with the given Process Flow in the Outbound table. This will display the `Configuration Login` dialog box as shown in Figure 182.



User Name default

Password ********

Application default

Host colorado

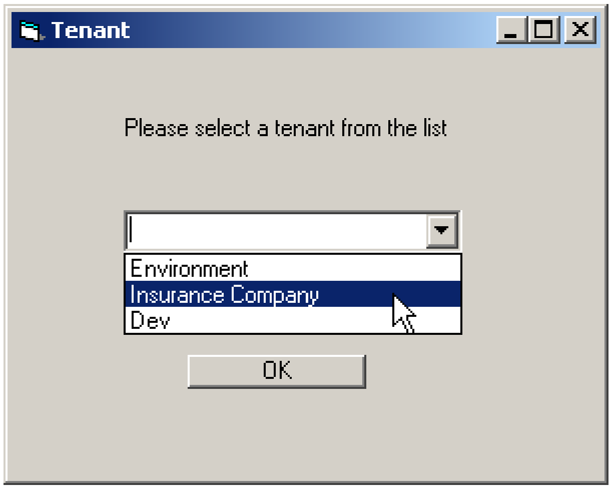
SOAP Port 2025

Figure 181:Configuration Login Dialog Box

7. Enter the following information and click `OK`:
   - `User Name`—User name to log in to Configuration Layer
   - `Password`·Password that is associated with this user name
   - `Application`—Name of the application to use in the Configuration Layer.
   - `Host`—Host name of Configuration Server.
   - `SOAP Port`—SOAP port number that is used to communicate with the Configuration Server.

Note: This is not the Configuration Server's primary port number that is used by other applications, such as Configuration Manager.

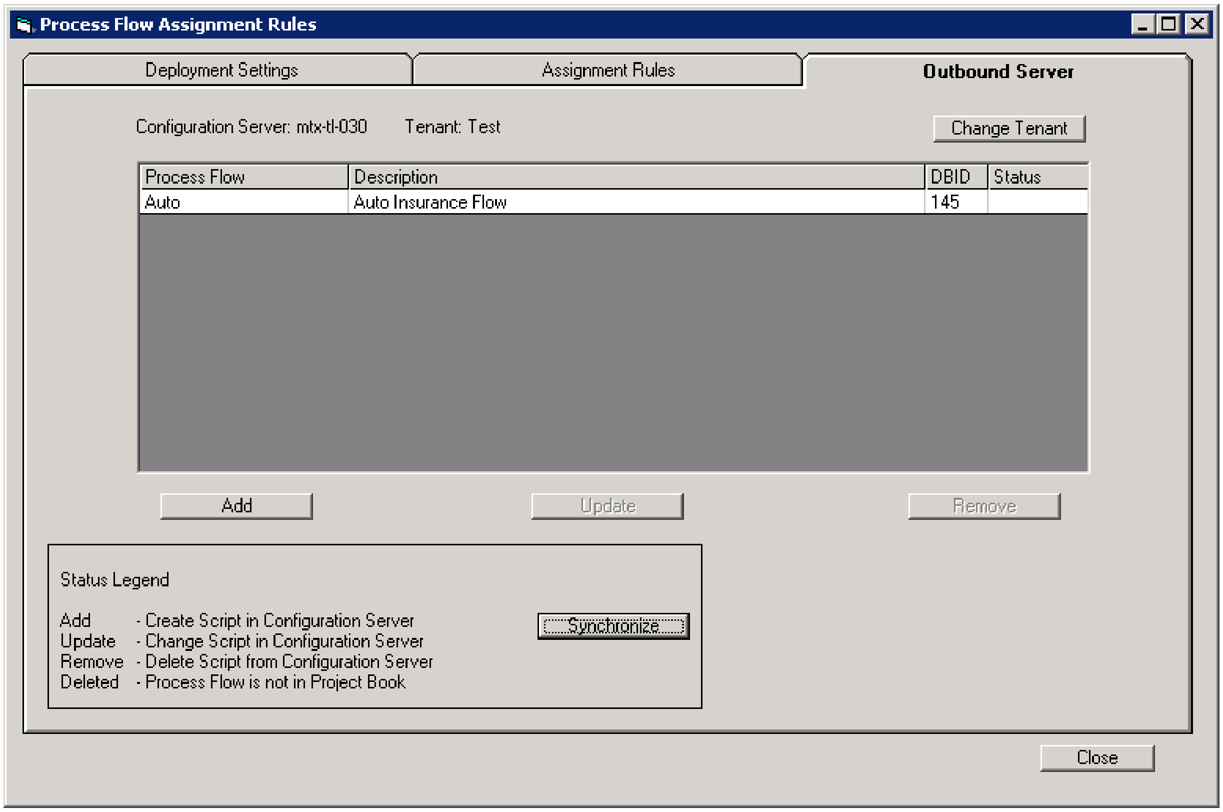
8. If a tenant name exists in the upper left-hand corner of the Outbound Server tab, that tenant will be used for the `Synchronize` operation. Otherwise, the `Tenant` dialog box will be displayed. Select a tenant from the dropdown list as shown in Figure 182 and click `OK`.



Please select a tenant from the list

Figure 182:Select Tenant

The DBID column in the Outbound table is updated to reflect the Script object that is created in Configuration Server as shown in Figure 183. If needed, the Tenant label is also updated with the current working tenant.



Configuration Server: mtx-tl-030 Tenant: Test

Process Flow	Description	DBID	Status
Auto	Auto Insurance Flow	145	

Status Legend

Add	- Create Script in Configuration Server
Update	- Change Script in Configuration Server
Remove	- Delete Script from Configuration Server
Deleted	- Process Flow is not in Project Book

Figure 183:Outbound Server Tab - DBID Updated

End of procedure

Procedure: Associating a Script object to an Outbound Campaign Group, Campaign, or Calling List

Start of procedure

1. Log in to the Configuration Server using Configuration Manager.
2. Bring up the properties of the object to be modified: Campaign Group, Campaign, or Calling List.
3. Look for the Script dropdown list on the `Properties` dialog box and use the browse capability to associate the Script object with this Campaign Group, Campaign, or Calling List.

   When the script is specified in multiple Outbound Objects related to a particular record, then Outbound Contact Server (OCS) selects the script DBID in the following order:

   a. Campaign Group
   b. Campaign
   c. Calling List

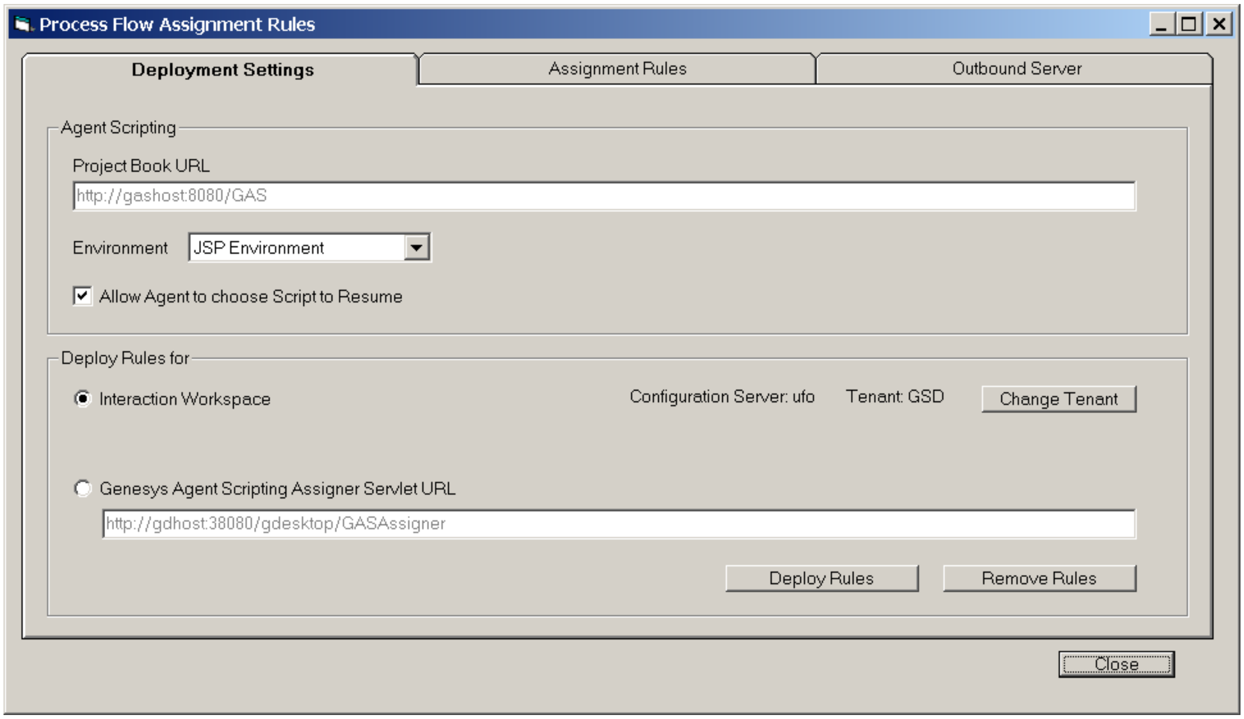
End of procedure

For more information, consult the *Outbound Contact* 8.0 *Reference Manual*, “Assignment Rules Deployment,” with two subsections: Genesys Desktop (Assigner Servlet) and Interaction Workspace.

Deploying Assignment Rules

Assignment Rules are used to determine which deployed Process Flow to launch, based on the data associated with a Genesys interaction. Select `Assignment Rules` from the `Compile` menu to display the Process Flow Assignment Rules dialog box, where you can configure these rules. The Assignment Rules tab enables you to specify exactly how the integration with your custom launching process or Genesys product will work.

Assignment Rules are deployed as the Assigner Servlet for Genesys Agent Desktop server, and they are deployed as Scripts in the corresponding tenant of Genesys Configuration Server for Interaction Workspace.



Project Book URL: http://gashost:8080/GAS

Environment: JSP Environment

☑ Allow Agent to choose Script to Resume

● Interaction Workspace Configuration Server: ufo Tenant: GSD

○ Genesys Agent Scripting Assigner Servlet URL: http://gdhost:38080/gdesktop/GASAssigner

Figure 184:Process Flow Assignment Rules Dialog Box—Deployment Settings Tab

Table 77 describes the parameters that appear on the `Deployment Settings` tab of the `Process Flow Assignment Rules` dialog box.

Table 77: Deployment Settings Tab Parameters

Control	Description / What to Do...
Agent Scripting area	
Project Book URL field	Specifies the URL that is used by the Assigner to launch the selected Process Flow. If this URL is invoked without a specifying Process Flow, the default Process Flow for this Project Book will be launched. To set the value of this field, select the Target Environment from the `Environment` drop-down list.
Environment drop-down list	Select the Target Environment from this drop-down list.
Allow Agent to choose Script to Resume check box	Selectto enable the agent to select the custom Process Flow that is launched when a suspended script is resumed. If this check box is not selected, the previously suspended script will be resumed (or a new one will be launched, if the suspended script cannot be found).
Deploy Rules for area	

Table 77: Deployment Settings Tab Parameters (Continued)

Control	Description / What to Do...
Configuration Server label	Name of the current Configuration Server.
Tenant label	Tenant is a label showing the current working tenant where Assignment Rules and Outbound Campaign script objects are stored. An association with a tenant is made after a Change Tenant, Deploy Rules, Remove Rules, or Synchronize operation has been performed. If the association have not been made, the value is absent.
Change Tenant button	Click Change Tenant to set Configuration Server and Tenant where Assignment Rules for Interaction Workspace and Outbound Campaign script objects will be deployed. Clicking this button opens the Configuration Login dialog box, where you enter the user name, password, application, host, and SOAP port to connect to Configuration Server. After a successful connection to Configuration Server, a dialog box that contains the list of valid tenants is displayed. You can select a tenant from the list. The Tenant label is also updated with the name of the tenant that you select.
Interaction Workspace radio button	Select this radio button to deploy Project Book Assignment Rules by using the new deployment mechanism for Interaction Workspace. **Note:** If you see the message `Mark Done Support feature is disabled...` in red text to the right of this radio button, you can respond to it in the Advanced tab of the Setting dialog box—select the `Support Mark Done Feature` checkbox there.
Genesys Agent Scripting Assigner Servlet URL	Select this radio button to deploy the Assigner Servlet into your Genesys Agent Desktop Customization Server. **Note:** The Genesys Agent Desktop Customization Server can be any Java-based web server; in most cases, it is GAD server by itself.)
Assigner Servlet URL field	Specify the Assigner Servlet's location (as a URL).

Table 77: Deployment Settings Tab Parameters (Continued)

Control	Description / What to Do...
Remove Rules button	Click to remove all association of this Project Book from the Assigner Servlet.
Deploy Rules	Click to add or update the association of this Project Book with the Assigner Servlet. (Required to make the defined assignment rules active.) **Note:** Clicking this button opens a dialog box that shows the following question: `Should this Project Book be the default?` If you click `Yes`, the current Project Book and current script become the default for the Interaction Workspace Plug-In for Genesys Agent Scripting. The default Project Book is the one that will be used in the event that an interaction is encountered that does not have enough data associated with it to determine the correct Project Book to use. There will always be one—and only one—default Project Book that is known to the Assigner Servlet.

The Assigner Servlet uses data that is associated with the interaction (attached data, customer data, and/or agent data) to determine the appropriate Script to launch.

Note: Scripts can be launched without the Assigner Servlet As long as the following parameters are provided (`G_Interaction_ID` or `G_OR_ID`, and `G_Config_Place`), all Genesys Integration functionality will be accessible.

The Assigner Servlet uses the following predefined KV pairs that are attached to the Interaction to determine the correct Project Book and Process Flow to launch:

Table 78: Deployment Settings Tab Parameters

KV Pair	Description
`GAS_ScriptURL`	The deployed URL of the Project Book.
`GAS_ScriptID`	The ID of a previously launched Script Instance.
`GSW_Script_ID`	The `DB_ID` of the Script object that is associated with an Outbound call.

Table 78: Deployment Settings Tab Parameters (Continued)

KV Pair	Description
GAS_ScriptName	The specific Process Flow name that is contained in a Project Book.
GAS_RelationshipID	The ID that associates multiple Script Instances. This is used to allow an agent to query for the correct Script Instance to relaunch from a subset of previously suspended Script Instances.

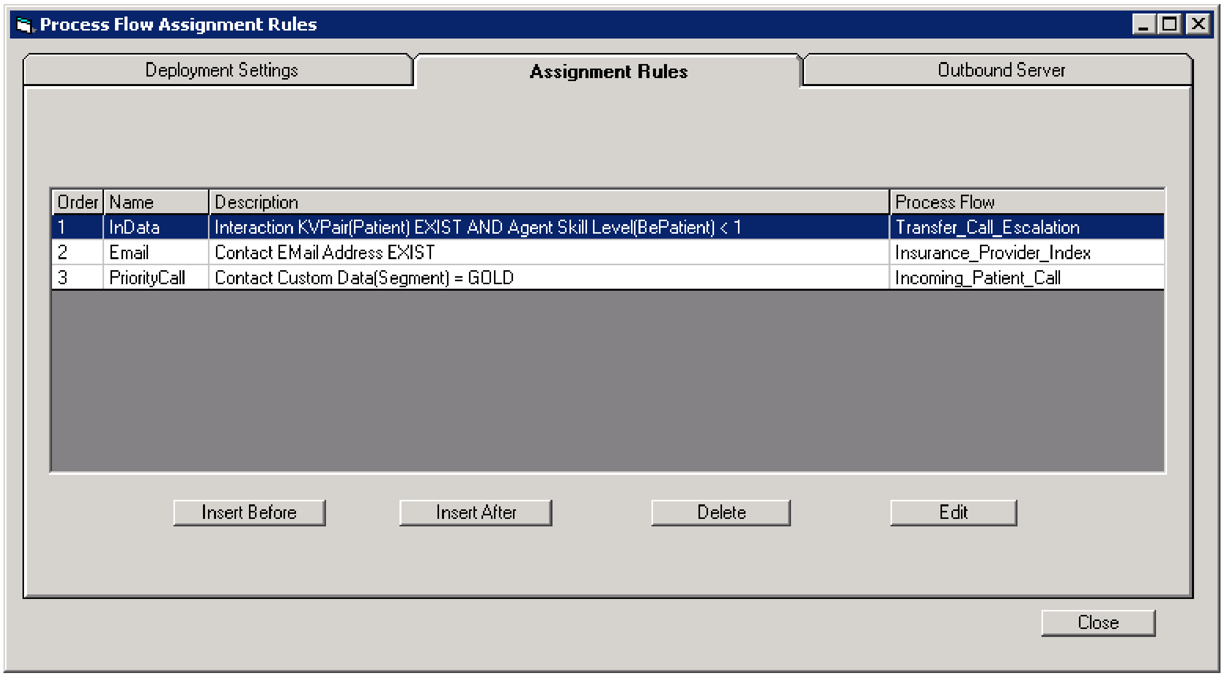
The priority that the Assigner Servlet will use to select the Process Flow is as follows:

1. **If GAS_ScriptURL and GAS_ScriptID are specified**
   The previously suspended Script Instance (indicated by GAS_ScriptID) is resumed, or a new Script Instance for the default Process Flow is launched (if the Script ID cannot be found).
2. **If GAS_ScriptURL and GSW_ScriptID are specified**
   A new Outbound call is being launched, and a new Script Instance is created for the Process Flow that is associated with the DB_ID defined on the Outbound tab of the Process Flow Assignment Rules dialog box. If the Process Flow association is not found, continue with Step 3.
3. **If GAS_ScriptURL and GAS_ScriptName are specified**
   A new Script instance is created for the Process Flow = GAS_ScriptName (if this Process Flow is undefined, the Default Process flow for the Project Book that is associated with the GAS_ScriptURL will be launched instead).
4. **If GAS_ScriptURL only is specified**
   The custom Assignment Rules from the Assignment Rules tab of the Process Flow Assignment Rules dialog box will be used (see Figure 185). The data that is associated with the interaction will be interrogated. If a condition is met, a new Script Instance for the Process Flow that is defined for that rule will be launched. If no condition is met, a new Script Instance for the default Process Flow for the Project Book will be launched.

   Note: You can define conditions by using the Process Flow Assignment Condition dialog box. See “Process Flow Assignment Conditions Dialog Box” on page 397.

5. **If GAS_ScriptURL is not specified**
   A new Script Instance for the default Process Flow for the default Project Book will be launched.

Note: For all of the preceding conditions under which a new Script instance is being created (with the exception of Item 1), if the `GAS_RelationshipID` parameter is specified and the `Allow Agent to Choose Script to Resume` check box is selected as part of the deployment settings (see Figure 184), the agent will be presented with a screen that contains a list of all previously suspended Script Instances that are associated with that Relationship ID. The agent can then choose which one to resume or, instead, create a new Script Instance.



Order	Name	Description	Process Flow
1	InData	Interaction KVPair(Patient) EXIST AND Agent Skill Level(BePatient) < 1	Transfer_Call_Escalation
2	Email	Contact EMail Address EXIST	Insurance_Provider_Index
3	PriorityCall	Contact Custom Data(Segment) = GOLD	Incoming_Patient_Call

Figure 185:Process Flow Assignment Rules Dialog Box— Assignment Rules Tab

When a Script is launched, the following parameters are provided as part of the Query string:

Table 79: Deployment Settings Tab Parameters

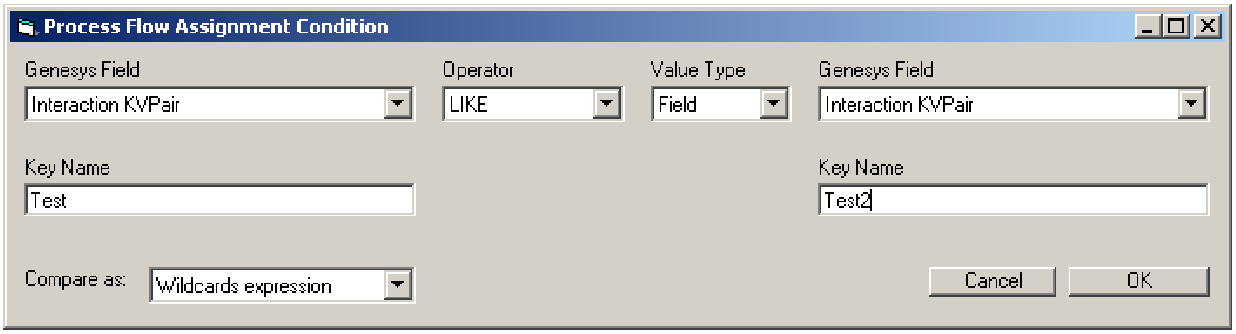
KV Pair	Description
`G_Interaction_ID`	Required if `G_OR_ID` is not specified. The T-Server Connection ID or AIL Interaction ID.
`G_Agent_ID`	The agent ID.
`G_Config_Place`	Required. The Genesys Place that is configured for this agent.

Table 79: Deployment Settings Tab Parameters (Continued)

KV Pair	Description
G_Contact_ID	The ID of the contact (Customer), as specified by MCR (Multimedia) Universal Contact Server.
G_Script_ID.	The ID of the Script instance, if a previously suspended script is to be resumed.
G_Script_RelationshipID	The Relationship ID that is used to associate previously suspended Script Instances.
G_Script_Query_Relationship	Either 0 or 1. When set (1), indicates that the agent should be presented with a list of previously suspended Script Instances that contain the specified Relationship ID. Not used if G_Script_RelationshipID is not specified.
G_OR_ID	Required if G_Interaction_ID not specified. The ID of the Outbound Record, used for Outbound calls only.
G_Assigner_URL	The location of the Assigner Servlet, specified as a URL. Necessary for the Assigner Servlet and the Script Instance to maintain synchronization.

Process Flow Assignment Conditions Dialog Box

The Process Flow Assignment Condition dialog box enables you to compare a Genesys field against another field or against a specific value. The details of this comparison will change according to the selections that you make for certain fields. For example, if you select Field as a Value Type, the GUI will appear similar to what is shown in Figure 186:



Process Flow Assignment Condition

Genesys Field: Interaction KVPair

Operator: LIKE

Value Type: Field

Genesys Field: Interaction KVPair

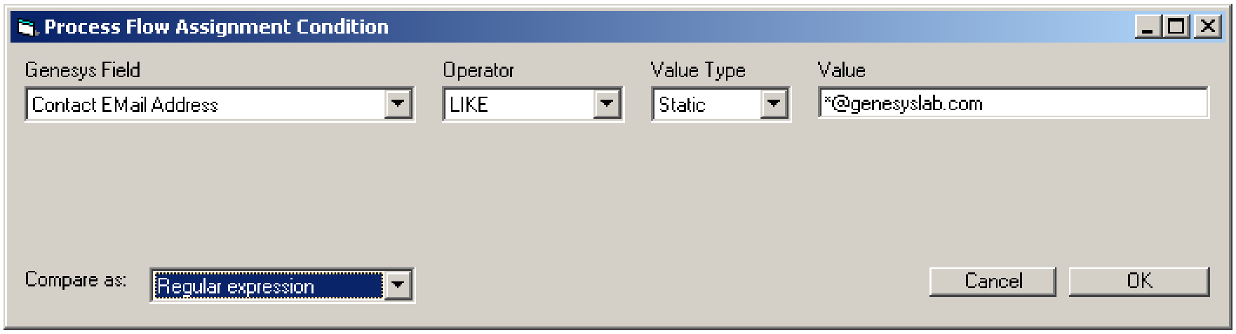
Key Name: Test

Key Name: Test2

Compare as: Wildcards expression

Figure 186:Process Flow Assignment Conditions Dialog Box—Field Value Type

However, if you select Static as a Value Type, the GUI will appear similar to what is shown in Figure 187.



Process Flow Assignment Condition

Genesys Field: Contact EMail Address

Operator: LIKE

Value Type: Static

Value: *@genesyslab.com

Compare as: Regular expression

Figure 187:Process Flow Assignment Conditions Dialog Box—Static Value Type

Table 80 describes the parameters that appear in the `Process Flow Assignment Condition` dialog box.

Table 80: Process Flow Assignment Condition Parameters

Parameter	Description
Genesys Field	Select the appropriate Genesys Field for this condition from the drop-down list. Valid values are: • `Interaction KVPair.` • `Agent Skill Level.` • `Contact e-mail Address.` • `Contact Phone Number.` • `Contact Custom Data.`

Table 80: Process Flow Assignment Condition Parameters (Continued)

Parameter	Description
Operator	Select the appropriate operator for this condition from the drop-down list. Valid values are: • > (Greater than)—The left operand is greater than the right operand. • < (Less than)—The left operand is less than the right operand. • = (Equal to)—The left operand is equal to the right operand. • != (Not equal to)The left operand is not equal to the right operand. • >= (Greater than or equal to)—The left operand is greater than or equal to the right operand. • <= (Less than or equal to)—The left operand is less than or equal to the right operand. • NOT EXIST—Can be applied to fields of type Interaction KVPair, Contact Custom Data, or Agent Skill Level. When this operator is used, the right operand is ignored. This operator checks for the existence of the corresponding key or agent skill level in the left operand. • EXIST—Can be applied to fields of type Interaction KVPair, Contact Custom Data, or Agent Skill Level. When this operator is used, the right operand is ignored. This operator checks for the existence of the corresponding key or agent skill level in the left operand. • LIKE—The left operand matches the expression in the right operand. • NOT LIKE—The left operand does *not* match the expression in the right operand.
Value Type	Select Field or Static from the drop-down list. If you select Field, the Genesys Field field appears on the right-hand side of the dialog box. The Genesys Field field will become the right operand in the comparison. If you select Static, the Value field will appear on the right-hand side of the dialog box. The Value field will become the right operand in the comparison.

Table 80: Process Flow Assignment Condition Parameters (Continued)

Parameter	Description
Genesys Field	The Genesys Field field appears when the Value Type parameter is set to Field. Valid values are: • Interaction KVPair • Agent Skill Level • Contact Email Address • Contact Phone Number • Contact Custom Data This field is the right operand in the comparison.
Value	The Value field appears when the Value Type parameter is set to Static. This field is the right operand in the comparison.
Key Name	Enter a Key Name for the condition. This text box can appear as both a left and a right operand in the condition if the corresponding Genesys Field drop-down lists are set to either Interaction KVPair or Contact Custom Data.

Table 80: Process Flow Assignment Condition Parameters (Continued)

Parameter	Description
Skill Name	Enter a Skill Name for the condition. This text box can appear as both a left and a right operand in the condition, if the corresponding Genesys Field drop-down lists are set to Agent Skill Level.
Compare as	Use this field to specify how the comparison will be conducted. Valid values are: • Default—Bases the comparison on the Genesys Field type (integer or string) of the left operand. If the field type is integer, the right operand is transformed into an integer, and a comparison of integers is performed. Likewise, if the field type is string, the right operand is transformed into a string. Valid operators are =, !=, >, <, NOT EXIST, and EXIST. • String—The value of each operand is transformed into a string, and the resulting strings are compared. This might produce an unexpected result; for example, if the left operand is 9, the right operand is 10, and operator is <, the result of comparison is FALSE (because, lexicographically, the string 9 is greater than the string 10). Valid operators are the same as described earlier for Default. • Integer—The value of each operand is transformed into an integer, and the resulting integers are compared. In this case, the example of 9<10 compares numbers (not strings); therefore it would evaluate to TRUE, unless one of the operands cannot be transformed to an integer. Valid operators are the same as described earlier for Default. • Wildcards expression—You can enter wildcard symbols into the right operand: ***** (asterisk) represents multiple characters, and **?** (question mark) represents a single character—for example, as in *.genesyslab.com or ?jones@genesyslab.com. Valid operators are LIKE and NOT LIKE. • Regular expression—You can enter a regular expression into the right operand—for example, \w*.genesyslab.com. Valid operators are the same as described earlier for Wildcards expression.

Integration With Genesys Software

The Interaction Workspace Plug-In for Genesys Agent Scripting enables integration of custom scripts (web applications) in Interaction Workspace. The plug-in renders agent scripts inside the interaction window of Interaction

Workspace, which is displayed as a tab that is similar to Contact, Responses, and others.

Interaction Workspace Plug-In depends upon your Interaction Workspace deployment type and is described in detail in the *Interaction Workspace Deployment Guide*.

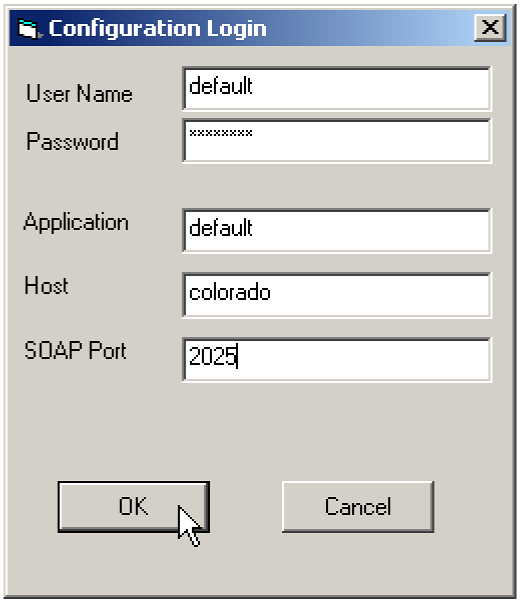
Interaction Workspace Rules Deployment

Assignment Rules for Interaction Workspace are stored as a Script object in the corresponding tenant of the configuration. The Interaction Workspace Plug-In for Genesys Agent Scripting is responsible for the assigning of Process Flow according to these rules.

Procedure: Deploying Assignment Rules for Interaction Workspace

Start of procedure

1. Click the **Deploy Rules** button to deploy assignment rules into Configuration Server.
2. Click `Synchronize` to create a script in Configuration Server that is associated with the given Process Flow in the table. This will display the `Configuration Login` dialog box as shown in Figure 188.


Configuration Login

User Name: default

Password: ********

Application: default

Host: colorado

SOAP Port: 2025

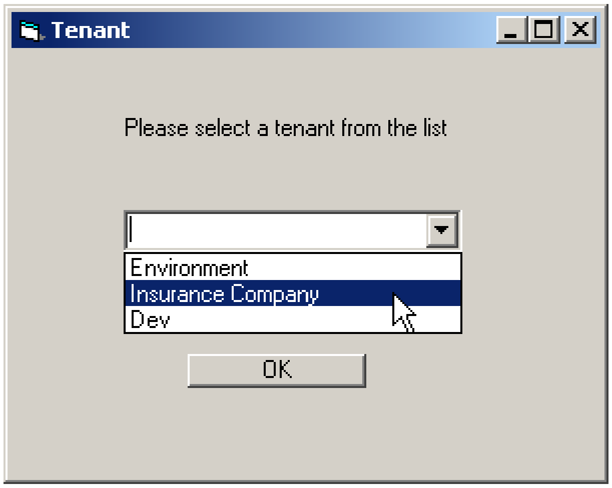
Figure 188:Configuration Login Dialog Box

3. Enter the following information and click `OK`:

- User Name—User name to log in to the Configuration Layer
- Password—Password that is associated with this user name
- Application—Name of the application to use in the Configuration Layer
- Host—Host name of Configuration Server
- SOAP Port—SOAP port number that is used to communicate with Configuration Server

Note: This is not the Configuration Server's primary port number that is used by other applications such as Configuration Manager.

4. If a tenant name exists in the upper left-hand corner of the Server tab, then that tenant will be used for the Synchronize operation. Otherwise, the Tenant dialog box will be displayed. Select a tenant from the dropdown list as shown in Figure 189 and click OK.



Tenant

Please select a tenant from the list

Environment
Insurance Company
Dev

Figure 189:Select Tenant

End of procedure

Genesys Agent Desktop Rules Deployment

The Rules Assigner for Genesys Desktop is deployed as Assigner Servlet, which must reside in a Java-based web server that is also running the Genesys Agent Interaction Layer.

The Assigner Servlet is used by Genesys Agent Desktop and can be used by other custom desktop or script-launching processes. Sample code is provided with the Genesys Agent Scripting installation to allow for easy integration. Refer to the *Genesys Agent Scripting 8.1 Deployment Guide* for additional details.

Genesys Agent Scripting Assigner Servlet Methods

When you are integrating with Genesys Agent Desktop (GAD), the Genesys Agent Scripting installation provides sample code that can be used directly in the GAD customization.

When you are integrating the Genesys Agent Scripting Assigner Servlet in other environments, two methods that are provided by the Assigner Servlet must be used:

- `public static GASAssigner getInstanceOf(String path)`

  Where `path` is the installed path of the Assigner Servlet, in which the deployed Agent Scripting Assignment Rules XML file will be deployed.

  A reference to the Genesys Agent Scripting Assigner Servlet (`GASAssigner`) will be returned. The Assigner Servlet is implemented as a Java singleton, so that the `getInstanceOf` method (and not its constructor) should be used.

  Example:

  ```
  GASAssigner assigner = GASAssigner.getInstanceOf("\\gdesktop");
  ```

- `public String getAssignedPFURL(String ailID) throws Exception`

  Where `ailID` is the AIL Interaction ID for the requested interaction.

  A string that represents the URL that should be invoked will be returned, with all required parameters defined. The user may then attach any additional parameters that are needed for the user's specific application (see the installed file, for additional sample usage):

  `·GenesysIntegration\gdesktop\custom\AgentScripting.jsp`

  Example:

  ```
  String ailIxnID = (String) request.getParameter
  ("GAD_InteractionID");
  assignedPFURL = assigner.getAssignedPFURL(ailIxnID);
  ```

An example URL that is generated by the Assigner Servlet would resemble the following:

```
http://dstevens:8080/IntegrateTest/WWGLaunch.jsp?WWGProcessFlowName=Gen
esys_Information&G_Interaction_ID=006701399027b008&G_Agent_ID=dstevens&
G_Config_Place=dstevens&G_Contact_ID=0000Aa10HJG100AX&G_Script_ID=&G_Sc
ript_RelationshipID=&G_Script_Query_Relationship=0&G_OR_ID=&G_Assigner_
URL=http%3a//colorado:8080/gdesktop/GASAssigner
```

The user may add additional parameters in the following form:

```
&name1=value1&name2=value2...
```

Chapter

16 Glossary of Key Terms

This chapter provides a Glossary of Key Terms related to Genesys Agent Scripting and the Genesys Agent Scripting Development Environment.

Action

An Action is a function that is performed when a button is clicked, when a Page is loaded or unloaded, or when a Page Layout is loaded.

Action List

The `Action List` shows all the Actions defined in the open Project Book and appears on the lower-right of the main Genesys Agent Scripting window in the default view.

Action Wizard

The Action wizard allows you to set up most Action types in a step-by-step fashion.

Branch

Branches are a set of logic that directs the flow from one Page to the next within the various Process Flows.

Branching can occur between any two Pages defined in the Project Book.

The `Branch` menu, accessed by right clicking a Branching Condition in the `Page Tree`, is used to insert, edit, or delete branches.

Catalog

A Catalog is where the page navigation information is stored. It is a list of links that appears as a navigational tool in the web browser when a script application runs. The Catalog provides a way to move to a Page without following the normal path of an interaction. The Page could be in the same Stream that is currently running, but it does not have to be. In fact, the Page referenced in a Catalog can be from different Streams and Process Flows entirely.

Data Validation

This refers to the concept of checking to see if the script user has entered valid data into a Field. Agent Scripting performs a level of data validation through the use of the `Format` field on the `Size` tab of the `Field Properties` dialog box.

Database

This refers to an external MS SQLServer, or Oracle database that a Genesys Agent Scripting application can read from or write to.

The database contains the data upon which scripts act when executed. You can access multiple databases when scripts are executed. These databases are the ones referenced in `Define` > `Databases` and `Define` > `Database Interfaces.`

Database Connection

Genesys Agent Scripting requires Database connection information in order to be able to read from or write to an external database. You define the parameters of a Database connection using `Define` > `Databases` and entering the required information.

Database Interface

A database interface allows you to access external databases to read or write data.

It provides easy access to back-end systems and their associated information. Data can be read from or written to the back-end system automatically. The data remains available throughout the interaction without any action on the part of the agent.

In order to retrieve information that is stored in an external database, or update the information that is stored in a database table, you need to create a Database Interface in the Agent Scripting Development Environment.

A Database Interface functions as a *query* to a specific external database. Genesys Agent Scripting supports three types of queries:

- **Select** — Retrieves information from an external database.

- **Update** — Inserts new rows, updates existing rows, or deletes rows in a database table. In order to insert rows, your database will need to have a key column defined.
- **Stored Procedure** — Passes values as input parameters to a stored procedure and maps output parameters and results (return parameters) to Fields.

Field

A Field is any data value gathered or stored by Genesys Agent Scripting. This includes information that you enter into a form on a web page, or data gathered from outside applications.

Once a value has been stored in a Field, it remains in the Field even if you move on to another Page. Data remains available throughout the entire session.

Field List

The `Field List` shows all the Fields defined in the open Project Book and appears on the upper-right of the main Genesys Agent Scripting window in the default view.

Import Objects

The `Import` feature of Genesys Agent Scripting allows you to import objects from other Project Books into the current Project Book. This is useful if you have defined Actions, Fields, Database Interfaces, Catalogs, Page Layouts, or other objects in a different Project Book and would like to make those objects available in the Project Book you are currently working on.

Menus (Menu Bar)

Genesys Agent Scripting menus organize all the features and functionality of the software.

The following menus are available in Genesys Agent Scripting:

- File
- Edit
- Project
- Process Flow
- Stream
- Page (or Page Layout)
- Format
- Define

- Compile
- Window
- Help

Many of the options available from menus are also available through pop-up menus. Simply right mouse click an item to display its pop-up menu.

Page

Pages are the individual web pages that the script users (agents) see when the script application runs. Pages can contain text, Fields to display or obtain data, Action buttons that trigger execution of code, or Action code that is executed when a Page loads or unloads as the script user navigates through the Stream.

Page Editor

The `Page Editor` is the main work area in Genesys Agent Scripting where you create the Pages for your application. It appears in the center of the main Genesys Agent Scripting window in the default view.

The `Page Editor` shows the content of individual web pages in a Stream. The `Page Editor` for a particular Page is accessed by clicking the desired Page in the `Page Tree.`

In the `Page Editor,` you can change information, add text, delete text, cut and paste text or images, add Fields and Actions from the `Field List` and `Action List,` and edit HTML code if you are comfortable with HTML.

Page Layout

A Page Layout describes how Pages will appear to the script user in the production environment. Page Layouts create a unique “look and feel” for scripts and provide an easy way to add an image or style to multiple Pages.

A Page Layout can include different graphical sections, backgrounds, and style sheets. You can create multiple Page Layouts and use them within a single Process Flow or Stream.

Page Layout Editor

The area occupied by the `Page Editor` becomes the `Page Layout Editor` when you select a Page Layout in the `File` menu. You can resize, move, or delete areas in the Page Layout, as well as add Fields, Actions, text, images, and a background using the `Page Layout Editor.`

Page Tree

The `Page Tree` shows a hierarchical list of all the defined Pages and Branching Conditions contained within the current Project Book. It appears on the lower-left of the main Genesys Agent Scripting window in the default view. If a Project or Process Flow is selected in the `Project Tree`, the `Page Tree` will be empty.

When you select a Stream in the `Project Tree,` you will see the Pages associated with that Stream in the `Page Tree.`

Icons appear next to the text at each level in the tree indicating what type of object is at each level.

Process Flow

A Process Flow represents a single business process from start to finish, such as signing up a new customer.

Project

A Project is a collection of Process Flows. Projects are high-level groupings of related business processes, for example, all processes associated with new customer acquisition.

Project Book

A Project Book is another name for the Genesys Agent Scripting database comprised of objects such as Projects, Process Flows, Streams, Pages, Fields, Actions, and Interfaces.

Genesys Agent Scripting stores the metadata (the information that describes the business process being created), in a Microsoft Access database (`.mdb` file).

Project Tree

The Project Tree shows a hierarchical list of all the defined Projects, Process Flows, and Streams contained within the current Project Book. It appears on the upper-left of the main Genesys Agent Scripting window in the default view.

Icons appear next to the text at each level in the tree indicating what type of object is at each level.

Settings

The `Settings` dialog box defines default parameters for a Genesys Agent Scripting Project Book. You set the default directory for images and style sheets, identify the default Page Layout and Process Flow, and configure other global settings for your Project Book.

Simple Object Access Protocol (SOAP)

Simple Object Access Protocol (SOAP) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.The framework has been designed to be independent of any particular programming model and other implementation specific semantics.

Stream

A Stream is a collection of Pages with a logical flow. Branching Conditions govern the transition from Page to Page.

Streams are subprocesses that are likely to be reused, such as a credit card validation subprocess, or a subprocess that updates a database in a specific way.

Style Sheet

Style sheets are files defined in the form of rules that tell a web browser how to display specific types of content structures when it encounters these structures in delivering the web page to a user.

Target Directory

A Target Directory is the directory to which the ASP, ASPX, or JSP code will be compiled.

Target Environment

The Target Environment specifies the target directory to which the code will be compiled, the type of code to generate, and other necessary information about your application’s development environment. You create or select a Target Environment using the `Compile` menu.

Target Platform

The Target Platform is the type of code that Genesys Agent Scripting will compile:

- Active Server Pages (ASP) or Active Server Pages - Extended (ASPX) to run on Microsoft IIS web servers
- Java Server Pages (JSP) to run on a Java-based web server such as Apache Tomcat.

Supplements

Related Documentation Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources as necessary.

Genesys Agent Scripting

- *Genesys Agent Scripting 8.1 Help*, which explains how to use Genesys Agent Scripting and its associated components.
- *Genesys Agent Scripting 8.1 Deployment Guide*, which explains how to install and configure Genesys Agent Scripting.
- Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at `http://genesys.com/customer-care`.

Genesys Agent Desktop

- *Genesys Agent Desktop 7.6 Deployment Guide,* which explains how to install and configure Genesys Agent Desktop and its associated components.

Interaction Workspace

- *Interaction Workspace 8.1 Deployment Guide,* an online wiki that explains how to install and configure Interaction Workspace and its associated components.

Genesys

- *Genesys Glossary,* which provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.

- *Genesys Migration Guide*, which ships on the Genesys Documentation Library DVD, provides documented migration strategies for Genesys product releases. Contact Genesys Technical Support for more information.

Information about supported hardware and third-party software is available on the Genesys Documentation website in the following documents:

- *Genesys Supported Operating Environment Reference Guidel*
- *Genesys Supported Media Interfaces Reference Manual*

Consult the following additional resources as necessary:

- *Genesys Hardware Sizing Guide,* which provides information about Genesys hardware sizing guidelines for the Genesys 8.x releases.
- *Genesys Interoperability Guide,* which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and Gplus Adapters Interoperability.
- *Genesys Licensing Guide,* which introduces you to the concepts, terminology, and procedures that are relevant to the Genesys licensing system.

For additional system-wide planning tools and information, see the release-specific listings of System Level Documents on the Genesys Documentation website.

Genesys product documentation is available on the:

- Genesys Technical Support website at `http://genesys.com/customer-care`.
- Genesys Documentation website at `http://docs.genesys.com/`.
- Genesys Documentation Library DVD which you can order by e-mail from Genesys Order Management at `orderman@genesys.com`.

Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

80fr_ref_06-2008_v8.0.001.00

You will need this number when you are talking with Genesys Technical Support about this product.

Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

Type Styles

Table 81 describes and illustrates the type conventions that are used in this document.

Table 81: Type Styles

Type Style	Used For	Examples
Italic	- Document titles - Emphasis - Definitions of (or first references to) unfamiliar terms - Mathematical variables Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 416).	Please consult the *Genesys Migration Guide* for more information. Do *not* use this value for this option. A *customary and usual* practice is one that is widely accepted and used within a particular industry or profession. The formula, $x + 1 = 7$ where x stands for . . .
Monospace font (Looks like `teletype` or `typewriter text`)	All programming identifiers and GUI elements. This convention includes: - The *names* of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages. - The values of options. - Logical arguments and command syntax. - Code samples. Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line.	Select the `Show variables on screen` check box. In the `Operand` text box, enter your formula. Click `OK` to exit the `Properties` dialog box. T-Server distributes the error messages in `EventError` events. If you select `true` for the `inbound-bsns-calls` option, all established inbound calls on a local agent are considered business calls. Enter `exit` on the command line.
Square brackets ([ ])	A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information.	`smcp_server -host [/flags]`
Angle brackets (< >)	A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise. **Note:** In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values.	`smcp_server -host <confighost>`

Genesys

Index