Genesys Desktop 7.6

Developer'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.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Technical Support 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

Genesys, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other trademarks and trade names referred to in this document are the property of other companies. The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

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 Technical Support at the following regional numbers:

Region	Telephone	E-Mail
North and Latin America	+888-369-5555 or +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44-(0)-1276-45-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Japan	+81-3-6361-8950	support@genesyslab.co.jp

Prior to contacting technical support, please refer to the *Genesys Technical Support 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.genesyslab.com

Document Version: 76gd_dev_dtop_06-2012_v7.6.401.00

Table of Contents

Preface

Welcome to the *Genesys Desktop 7.6 Developer's Guide.* This document introduces you to the concepts, terminology, and procedures relevant to customizing the Genesys Desktop 7.6 software with buttons, menus, and other features.

This document is valid only for the 7.6 release of this product.

This document is intended for use with both the *Genesys Desktop 7.6 XML Reference* and the *Genesys Desktop 7.6 Customization Examples* (see "Related Resources" on page 12).

Note: Genesys Desktop was known in 6.*x* releases as Genesys Contact Navigator (GCN).

For releases of this document created for other releases of this product, please visit the Genesys Technical Support website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.

This chapter contains the following sections:

Overview of Genesys Desktop 7.6

Genesys Desktop 7.6 is a web-based desktop application that contact center agents, supervisors, and knowledge workers can use to perform online

communication tasks. It is a sophisticated tool for communication between customers and companies, enabling interactions through the following media:

- E-mail
- Voice (featuring Web CallBack Request, Outbound Campaign Calls and Voice CallBack)
- Chat

Furthermore, Genesys Desktop includes a Supervisor component that allows supervisors to manage e-mail traffic, monitor the performance of your contact center resources, and configure agents.

Intended Audience

This guide is primarily intended for those who must configure Genesys Desktop 7.6. It assumes that you have a basic understanding of, and familiarity with:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications.
- Genesys Framework and related solutions.
- Network design and operation.
- Your own network configurations.

You should also be familiar with the use of Genesys Configuration Manager and Framework.

Usage Guidelines

The Genesys developer materials outlined in this document are intended to be used for the following purposes:

- Creation of contact-center agent desktop applications associated with Genesys software implementations.
- Server-side integration between Genesys software and third-party software.
- Creation of a specialized client application specific to customer needs.

The Genesys software functions available for development are clearly documented. No undocumented functionality is to be utilized without Genesys's express written consent.

The following Use Conditions apply in all cases for developers employing the Genesys developer materials outlined in this document:

1. Possession of interface documentation does not imply a right to use by a third party. Genesys conditions for use, as outlined below or in the *Genesys Developer Program Guide,* must be met.

2. This interface shall not be used unless the developer is a member in good standing of the Genesys Interacts program or has a valid Master Software License and Services Agreement with Genesys.
3. A developer shall not be entitled to use any licenses granted hereunder unless the developer's organization has met or obtained all prerequisite licensing and software as set out by Genesys.
4. A developer shall not be entitled to use any licenses granted hereunder if the developer's organization is delinquent in any payments or amounts owed to Genesys.
5. A developer shall not use the Genesys developer materials outlined in this document for any general application development purposes that are not associated with the above-mentioned intended purposes for the use of the Genesys developer materials outlined in this document.
6. A developer shall disclose the developer materials outlined in this document only to those employees who have a direct need to create, debug, and/or test one or more participant-specific objects and/or software files that access, communicate, or interoperate with the Genesys API.
7. The developed works and Genesys software running in conjunction with one another (hereinafter referred to together as the "integrated solutions") should not compromise data integrity. For example, if both the Genesys software and the integrated solutions can modify the same data, then modifications by either product must not circumvent the other product's data integrity rules. In addition, the integration should not cause duplicate copies of data to exist in both participant and Genesys databases, unless it can be assured that data modifications propagate all copies within the time required by typical users.
8. The integrated solutions shall not compromise data or application security, access, or visibility restrictions that are enforced by either the Genesys software or the developed works.
9. The integrated solutions shall conform to design and implementation guidelines and restrictions described in the *Genesys Developer Program Guide* and Genesys software documentation. For example:
    a. The integration must use only published interfaces to access Genesys data.
    b. The integration shall not modify data in Genesys database tables directly using SQL.
    c. The integration shall not introduce database triggers or stored procedures that operate on Genesys database tables.

Any schema extension to Genesys database tables must be carried out using Genesys Developer software through documented methods and features.

The Genesys developer materials outlined in this document are not intended to be used for the creation of any product with functionality comparable to any

Genesys products, including products similar or substantially similar to Genesys's current general-availability, beta, and announced products.

Any attempt to use the Genesys developer materials outlined in this document or any Genesys Developer software contrary to this clause shall be deemed a material breach with immediate termination of this addendum, and Genesys shall be entitled to seek to protect its interests, including but not limited to, preliminary and permanent injunctive relief, as well as money damages.

Chapter Summaries

In addition to this opening chapter, the *Genesys Desktop 7.6 Developer's Guide* contains the following chapters:

- Chapter 1, "About Genesys Desktop," on page 15, describes the Genesys Desktop architecture and provides an overview of its principal features.
- Chapter 2, "Agent Desktop Customization," on page 23, outlines the deployment issues related to customizing the Genesys Desktop solution, presents examples of code for adding a button, a menu, and a tab to the interface of your customized application, and contains the code for a template customization file with all tags deployed.
- Chapter 3, "Open Media Extensions," on page 79, outlines the extension procedures for the Genesys Desktop 7.6 GUI Layer.
- Chapter 4, "Navigation and Controls," on page 95, outlines the customization that keyboard navigation in custom pages and Genesys Desktop shortcuts in Genesys Desktop 7.6.
- Chapter 5, "Localization in JSP and Java Code," on page 117, outlines how to localize certain interface items such as buttons and tabs.
- Chapter 6, "Agent and Supervisor Desktop Interception API," on page 121, outlines the customization that enables server-side interception of agent actions in Genesys Desktop 7.6.
- Chapter 7, "Supervisor Desktop Customization," on page 133, describes how to add customer-specific data to the Supervisor Desktop views.

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:

gd_dev_dtop_10-2007_v7.6.000.01

You will need this number when you are talking with Genesys Technical Support about this product.

Type Styles

Italic

In this document, italic is used for emphasis, for documents' titles, for definitions of (or first references to) unfamiliar terms, and for mathematical variables.

Examples:

- Please consult the *Genesys Migration Guide* for more information.
- *A customary and usual practice* is one that is widely accepted and used within a particular industry or profession.
- Do *not* use this value for this option.
- The formula, $x + 1 = 7$ where x stands for . . .

Monospace Font

A monospace font, which looks like `teletype or typewriter text`, is used for 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; and code samples.

Examples:

- Select the `Show variables on screen` check box.
- Click the `Summation` button.
- In the `Properties` dialog box, enter the value for the host server in your environment.
- In the `Operand` text box, enter your formula.
- Click `OK` to exit the `Properties` dialog box.

- The following table presents the complete set of error messages T-Server® distributes 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.

Monospace is also used for any text that users must manually enter during a configuration or installation procedure, or on a command line:

Example:

- Enter `exit` on the command line.

Screen Captures Used in This Document

Screen captures from the product GUI (graphical user interface), as used in this document, may sometimes contain a minor spelling, capitalization, or grammatical error. 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.

Square Brackets

Square brackets indicate that a particular parameter or value is optional within a logical argument, a command, or some programming syntax. That is, the parameter's or value's presence is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information. Here is a sample:

```
smcp_server -host [/flags]
```

Angle Brackets

Angle brackets indicate a placeholder for a value that the user must specify. This might be a DN or port number specific to your enterprise. Here is a sample:

```
smcp_server -host <confighost>
```

Related Resources

Consult these additional resources as necessary:

- *Genesys Desktop Deployment Guide,* which ships on the Genesys Documentation Library DVD, and which outlines Genesys Desktop configuration and installation procedures.
- Code examples within this *Developer's Guide.*

- *Genesys Desktop 7.6 XML Tag Reference*, which provides information for the *Genesys Desktop 7.6 Customization Examples*. Both of these resources can be found on the Genesys Developer web site.
- *Multi-Channel Routing 7 Event Media Deployment Guide,* which introduces the architecture, required components, and procedures relevant to the deployment of a Genesys Multi-Channel Routing solution.
- *Genesys Voice Callback 7 Deployment Guide,* which provides configuration details for the Voice Callback components.
- The Apache documentation about Tomcat, connectors, and other Apache components is available on the Apache Foundation website at:
  - `http://www.apache.org`.
  - `http://jakarta.apache.org` (for Apache Foundation Java-platform projects).
- *Genesys Technical Publications Glossary,* which ships on the Genesys Documentation Library DVD, and which provides a comprehensive list of the Genesys and CTI terminology and acronyms used in this document.
- *Genesys Migration Guide,* also on the Genesys Documentation Library DVD, which contains a documented migration strategy for Genesys product releases 5.*x* and later. (Contact Genesys Technical Support for additional information.)
- *Genesys Licensing Guide,* which will help you understand the licensing process.
- Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at `http://genesyslab.com/support`.

Information on supported hardware and third-party software is available on the Genesys Technical Support website in the following documents:

- *Genesys Supported Operating Environment Reference Manual*
- *Genesys Supported Media Interfaces Reference Manual*

Genesys product documentation is available on the:

- Genesys Technical Support website at `http://genesyslab.com/support`.
- Genesys Developer website at `http://devzone.genesyslab.com`.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at `orderman@genesyslab.com`.

Making Comments on This Document

If you especially like or dislike anything about this document, please feel free to e-mail your comments to `Techpubs.webadmin@genesyslab.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 information in this document only and to the way in which the information is presented. Speak to 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.

Chapter

1 About Genesys Desktop

Genesys Desktop comprises two graphical user interfaces: Genesys Agent Desktop and Genesys Supervisor Desktop. This chapter describes the Genesys Desktop 7.6 architecture and the customization considerations associated with deploying the Agent Desktop and Supervisor Desktop functionality.

This chapter contains the following sections:

Overview

The Genesys Desktop application is a purely web-based application—it runs in a web browser—no components are installed on the client PC. It is ready to use. It does not require a lengthy installation and configuration process. Genesys Desktop features include:

- A single, unified graphical user interface (GUI) that contact-center agents and supervisors can use for all Genesys interactions.
- Customizable agent features.
- Customizable supervisor features.
- A toolkit of GUI objects for customizing agent desktops.

- An architecture that reduces the administrative burden associated with maintaining and upgrading application versions.

General Deployment Topology

Genesys Desktop runs on a web server and connects to the Genesys Framework solution. This architecture yields benefits in several areas:

- Communication—HTTPS (HyperText Transfer Protocol Secure) is used between the client application and the server.
- Security—Encrypted data exchange is available using the regular HTTPS mechanism provided by the web servers.
- Scalability—Load balancing is based on the regular load balancing functionality provided by the web server.

In Figure 1, the web server contains the Genesys Desktop components.

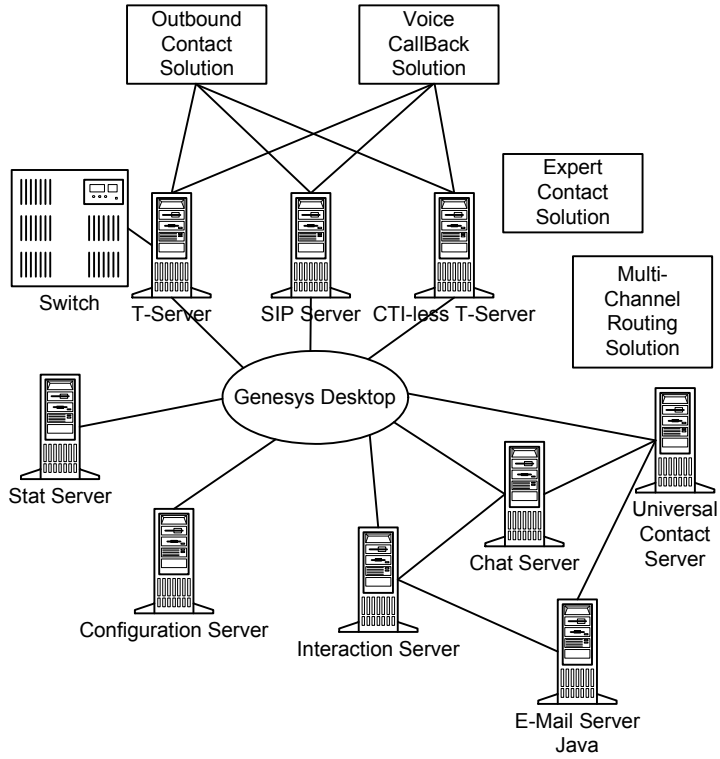


Figure 1: Deployment Topology

Genesys Desktop Application and Server Architecture

The Genesys Desktop application comprises two layers:

- Presentation Layer—Used for DHTML generation.
- Business Layer—A JavaBeans library that embeds some logic and presents an interface to all Genesys server components. (Genesys offers a separate Agent Interaction (Java) SDK product, which provides a JavaBeans–based agent desktop API.)

By default, the two layers that make up the Genesys Desktop application are packaged with the Apache web server and the Tomcat servlet engine. The Genesys Desktop application itself is deployed as a web application (see Figure 2).

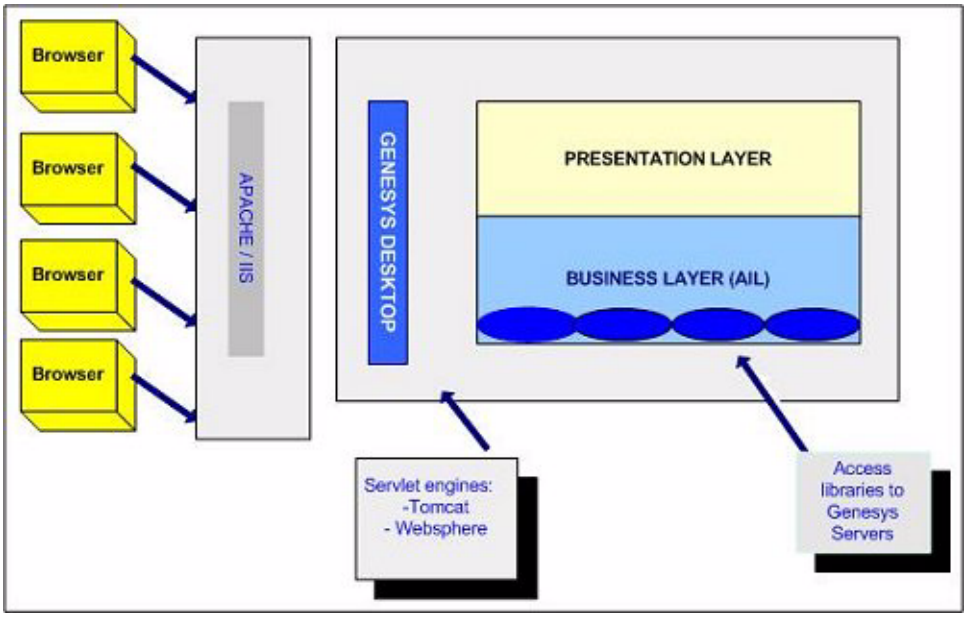


Figure 2: Genesys Desktop Architecture

There is no single Genesys Desktop servlet. The Genesys Desktop GUI component comprises several JSPs (Java Server Pages), servlets, and Java components. The purpose of the Genesys Desktop GUI is to produce the DHTML pages sent by the web server. The presentation layer does not store any information about the agent, interactions, and so on—it relies on the Business Layer for this. The Genesys Desktop GUI Layer focuses exclusively on GUI generation and GUI event-handling.

The servlet engine dynamically compiles JSPs to servlets. These pages are explicitly transformed in Java source files that implement servlets, and they are then compiled like regular servlets.

There are no particular limitations or restrictions applying to servlets or JSPs in the Genesys Desktop application.

Directory Structure

By default, Genesys Desktop is embedded in Tomcat and deployed as a web application. Therefore, it is installed in the appropriate Tomcat directory tree, in the `GenesysDesktop/` directory. Figure 3 shows a representative directory tree.

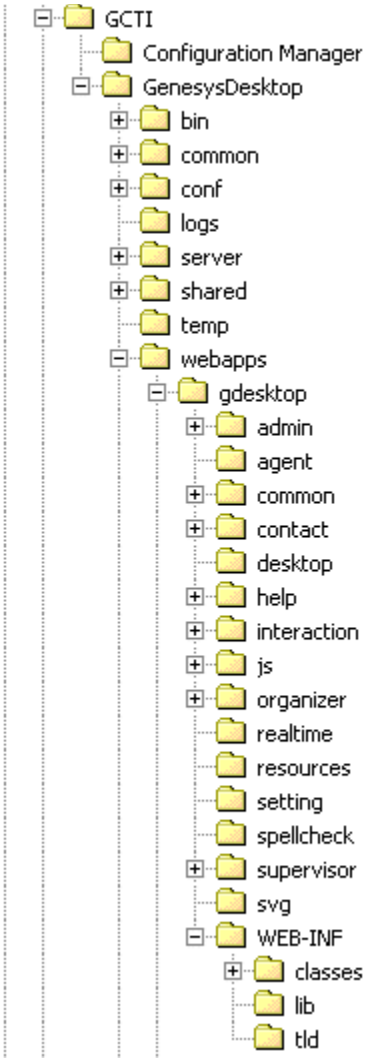


Figure 3: The Tomcat Directory Tree

You can also pack your servlet classes in a `JAR` (Java Archive) file. You then must copy the `JAR` file to the `WEB-INF/library/` directory.

Because you can write your own `jsp` files, you can also write your own servlets. You must copy the `jsp` files to the `GenesysDesktop/` directory, and you must copy the servlet class files to the `WEB-INF/classes/` directory.

In Figure 3 on page 19, the `webapps/` directory is the Tomcat web application directory. The `webapps/gdesktop/` subdirectory is the Genesys Desktop application directory.

In this document, unless otherwise specified, all paths are relative to the following application directory:

`GenesysDesktop/webapps/gdesktop/.`

For example, the full path of a customization file, `custom/custom.xml`, is:

`GenesysDesktop/webapps/gdesktop/custom/custom.xml.`

Agent Desktop Customization

You can customize the Genesys Agent Desktop GUI by adding:

- Menus to the `Main` menu bar.
- Buttons to the `Interaction` frame.
- Tabs to the `Customer Records` and `Resources` frames.

You can also capture JavaScript events that occur in the GUI, and add your own JavaScript code.

In order to customize Genesys Agent Desktop, you must edit an XML customization file. You can also customize it with JSP (Java Server Page) files, HTML/DHTML pages, Servlet classes, and dictionary files. For further details, see “Customization Examples” on page 31.

Open Media Extensions

For each Open Media interaction type that you define in the Configuration Manager, you can specify how to display this particular interaction type in Genesys Agent Desktop GUI by using an Open Media extension.

Open media extensions are XML files loaded at Genesys Desktop startup. They enable you to set up the following components of Genesys Agent Desktop GUI:

- The navigation bar.
- Action buttons for interactions.
- The interaction display in the `media view` and `detail` panels.

To define an interaction extension, you must create a new extension directory in the `gdesktop/extension` directory. Then, you add an `interaction.xml` file to the new directory you created.

For further details about extension tags, see Chapter 3, “Open Media Extensions,” on page 79.

Agent Desktop Interception API

Chapter 6, "Agent and Supervisor Desktop Interception API," on page 121, outlines the customization that enables server-side interception of agent actions in Genesys Desktop 7.6.

Navigation and Controls

Chapter 4, "Navigation and Controls," on page 95, outlines the customization that enables keyboard navigation in custom pages and Genesys Desktop shortcuts in Genesys Desktop 7.6.

All the Genesys Desktop customization samples employ the navigation and controls functionalities; however, the following customization samples focus on keyboard navigation:

- `sample6 - interaction-information`
- `sample7 - contact-ext`
- `sample9 - central panel`
- `sample11 - scheduling client notification`
- `sample12 - external knowledgebase`
- `sample13 - custom area`
- `open-media-extension\extension\sms-out`

The other samples employ the navigation and controls functionalities for shortcuts only, or to publish a message in the status bar.

The JavaScript file `htmlcontrol.js` is used by the following customizations to enable HTML in custom panels:

- `sample6 - interaction-information`
- `sample7 - contact-ext`
- `sample12 - external knowledgebase`
- `sample13 - custom area`
- `open-media-extension\extension\sms-out`

For more information, see "htmlcontrol.js" on page 100.

Supervisor Desktop Customization

Chapter 7, "Supervisor Desktop Customization," on page 133, presents five code examples that illustrate possible ways to customize the Genesys Supervisor Desktop 7.6 GUI.

Chapter

2 Agent Desktop Customization

This chapter outlines the customization procedures for the Genesys Desktop 7.6 GUI Layer. It also presents five code examples that illustrate possible ways to customize the Genesys Desktop 7.6 graphical user interface (GUI). Genesys recommends that you carefully examine these examples before you begin customizing Genesys Desktop. See *Genesys Desktop 7.6 XML Tag Reference* for detailed descriptions of the tags used in these examples. Finally, it presents an XML customization file, which you can use as a template.

Genesys Desktop supports multiple customizations in parallel to facilitate the deployment of external customization without impacting the current customization.

This chapter contains the following sections:

Customization Principles

This section outlines the customization procedures for the Genesys Desktop 7.6 GUI Layer.

This section contains the following sub-sections:

Overview

You customize the GUI Layer primarily by editing a custom XML file that Genesys Desktop loads at startup; in some cases, however, you also edit a JSP file or certain other files, such as HTML files. To simplify Genesys Desktop update releases and corresponding migration procedures, custom code is kept separate from Genesys Desktop display code.

You can customize the Genesys Agent Desktop GUI with the following items:

- Buttons
- Menus
- Tabs
- Toolbar
- Shortcuts
- Additional JavaScript code for the events or actions that you have established for any added button, menu, toolbar item, or tab.

The customization XML file describes the elements to be added. The Genesys Desktop Server loads this file when it starts up, and dynamically adds your custom code to the regular Genesys Agent Desktop code (as shown in Figure 4).

Genesys Desktop supports multiple customizations in parallel. Each added customization is defined in a `customFile<n>` parameter. See “Deploying Multiple Customizations” on page 26.

Custom files are loaded and inserted in JSP in the configured order (`customFile, customFile2, customFile<n>`). The same logic applies to graphical components such as buttons and tabs.

If a conflict occurs between two customizations, only the first customization is taken into account; the second customization is fully disabled. Examples of items that could be in conflict include buttons, tabs, menus or menu items, toolbar items with the same name. Conflicts occur if more than one customization uses the same tag names, such as `<status-bar>`, `<chat>`, `<email>`, and `<voice>`.

In the event of a conflict, a warning message is written to the Genesys Desktop logs.

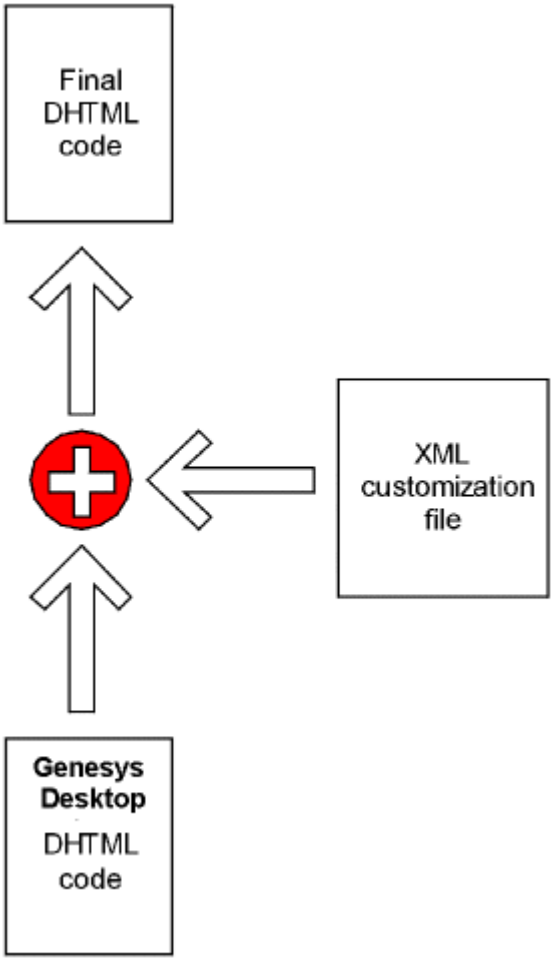


Figure 4: Dynamic Merging of Custom Code

For specifics about customizing Genesys Desktop, see Chapter 2, "Customization Examples," on page 31.

Deploying Your Custom Genesys Agent Desktop

To deploy your customized Genesys Agent Desktop solution:

1. In the `gdesktop/` application directory, create a subdirectory named `custom/`.
2. In the `custom/` directory that you just created, create a customization file named `custom.xml`.
3. In the configuration file `WEB-INF/web.xml`, specify the location of your customization file, so that the server is able to load it:
   a. Open the `WEB-INF/web.xml` file.
   b. Modify the text section shown in bold in the following excerpt:

Deploying a Single Customization

```
<servlet>
    <servlet-name>initUAD</servlet-name>
    <servlet-class>com.genesyslab.uadthin.InitUAD</servlet-class>
    <init-param>
        <param-name>configServerHost</param-name>
        <param-value>bre04-000038588</param-value>
    </init-param>
    <init-param>
        <param-name>configServerPort</param-name>
        <param-value>2020</param-value>
    </init-param>
    .
    <init-param>
        <param-name>customFile</param-name>
        <param-value>/custom/custom.xml</param-value>
    </init-param>
    .
</servlet>
```

Deploying Multiple Customizations

To deploy additional customizations in parallel without impacting the primary customization, specify the additional customizations by using the name `customFile<n>`, where `<n>` represents a customization number greater than integer 2. For example:

```
<init-param>
            <param-name>customFile</param-name>
            <param-value>custom/custom.xml</param-value>
</init-param>
<init-param>
            <param-name>customFile2</param-name>
            <param-value>custom2/custom.xml</param-value>
</init-param>
<init-param>
            <param-name>customFile3</param-name>
            <param-value>custom3/custom.xml</param-value>
</init-param>
```

c. Save the edited file.

> **Note:** You cannot use the space character for any information in the XML file.

4. Create any dictionary files that you need and place them in the `WEB-INF/classes/` directory. See “Dictionary Files” on page 27 for details.

> **Note:** All paths are case-sensitive, even on Microsoft Windows platforms.

Dictionary Files

A *dictionary file* is a collection of key-value pairs that link labels in various languages to keys inserted into your custom XML file. Dictionary files simplify localization of your customized Genesys Desktop GUI. Instead of entering the human-language text for custom buttons, menus, and tabs directly into your XML file, you enter a dictionary key that specifies which dictionary file Genesys Desktop uses to find the label text in the desired language. To use this feature, you must create a dictionary file for each language that your Genesys Agent Desktop GUI uses.

Note: No dictionary files are shipped with the Genesys Desktop product. You must create them.

Creating a Dictionary File

To create your default dictionary:

1. Enter your keys, as illustrated in the Example below.

   Each key identifies a custom object that you have created. The keys must be the same in each dictionary: only the values change in each dictionary.
2. Assign to each key its value in the desired default language.
3. Save this file as `custom.properties`.

   Dictionary files are stored in the `WEB-INF/classes/` directory. If necessary, create this directory.

   Note: Within the Genesys Desktop installation directory (shown as `GenesysDesktop/` in Figure 3 on page 19), the full path to the dictionary file is: `webapps/genesysdesktop/WEB-INF/classes/custom.properties`.

Example This example shows a dictionary file with English as the default language. The default language does not need to be English. Whatever values are attached to the keys saved in the default file are automatically the default values.

Note: The following keys are provided as examples only. Your welcome key could also be named `MyWelcomeKey`—provided that you always use the same key text in all dictionaries and in your custom XML file.

```
menu.welcome=Welcome!
menu.extend=Extend
menu.extend.contactaccount=Account
menu.extend.contactinfo=Contact Information
```

Creating Alternate Dictionary Files

To create a dictionary file in, for example, French:

1. Enter your keys, as illustrated in the Example below.

   The keys must be named the same in each dictionary. Only the values change in each dictionary.
2. Assign to each key its value in French.
3. Save this file as `custom_fr.properties` in the `/WEB-INF/classes/` directory.

   Note: Name your dictionary files using the default dictionary name plus an underscore and the correct ISO language code. Normally your corporate IP language.
4. Repeat this process for each additional language that you support, using the correct ISO codes in the file names.

Note: There is no limit to the number of dictionaries. However, each must include a valid ISO language code. You can add more dictionaries at any time, using this process.

Example of a French Dictionary File

```
menu.welcome=Bienvenue!
menu.extend=Extension
menu.extend.contactaccount=Compte du Contact
menu.extend.contactinfo=Information sur le Contact
```

Specifying the Root File Name

Specify in the Dictionary file the name of the default file, using the `<dictionary-class>` tag.

Specify only the root file name. Genesys Desktop deduces the other file names based on this root.

In the preceding examples, the root file name is `custom`.

Genesys Desktop understands the `.properties` extension and any attached ISO codes.

Selecting the Dictionary

At runtime, Genesys Agent Desktop registers the dictionary files. At this time, the user can select which dictionary to use.

Depending on which language Genesys Agent Desktop requires, it searches for the dictionary with the correct ISO code. If that code does not appear attached to any dictionary, Genesys Desktop uses the default dictionary.

Note: The dictionary file to be used is selected only at runtime. You do not specify this during program design.

Genesys Agent Desktop uses the values from the appropriate dictionary to label your custom objects in the desired language.

Customizable Features

Genesys Agent Desktop supports customization of several architectural features. You can add custom code to:

- DHTML pages.
- JSPs (Java Server Pages).
- Servlet classes.

You can use this custom code to call the Agent Interaction Layer, in order to retrieve more information regarding the interaction, or to take some action.

Genesys Desktop Calls the Agent Interaction Layer

When you use Genesys Agent Desktop, you have access to the Agent Interaction Layer (AIL) classes in JSPs or servlets. If you instantiate the `AilFactory` object, you access the same factory used in Genesys Desktop, which is a singleton (see Figure 5 on page 29).

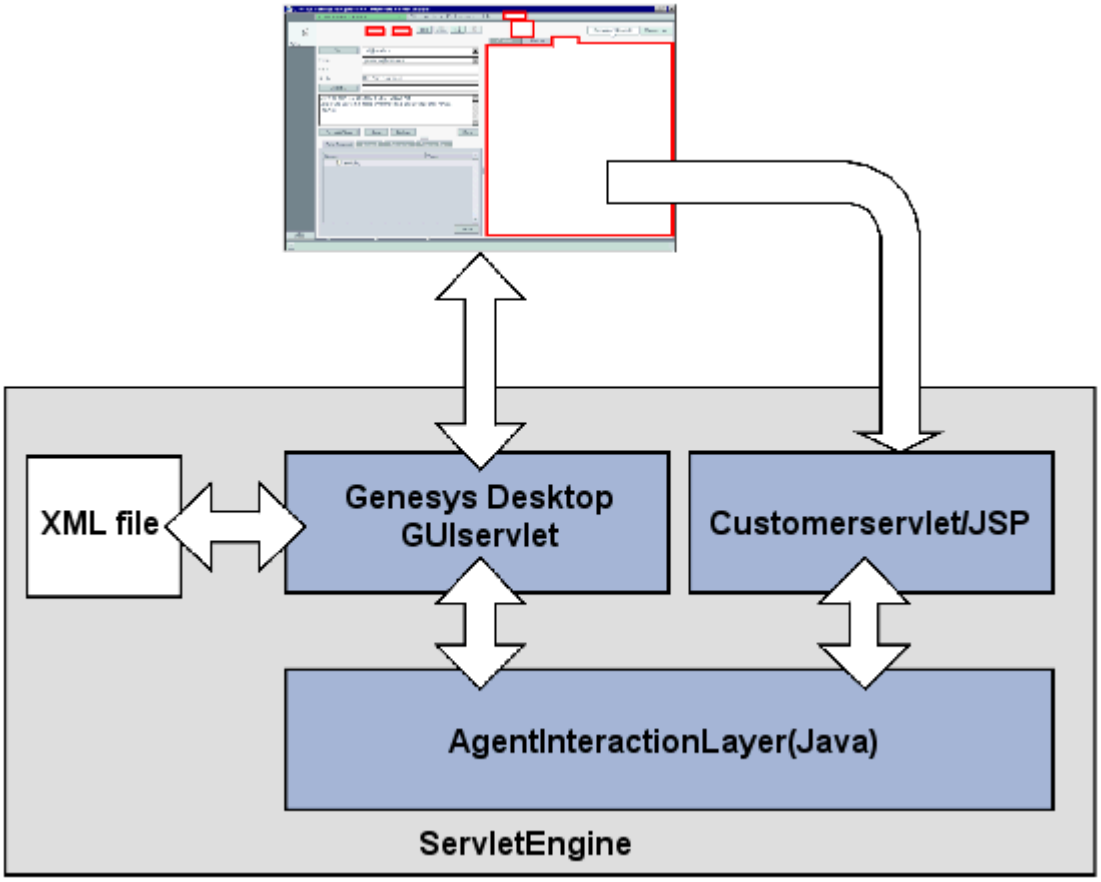


Figure 5: Genesys Desktop GUI Layer and Custom Code Accessing AIL

Therefore, to obtain a reference to the `AilFactory` object, enter in JSP code:

```
com.genesyslab.ail.AilFactory factory =
com.genesyslab.ail.AilLoader.getAilFactory();
```

This produces the same AIL instance as Genesys Desktop is using at runtime.

For more information about `AilFactory`, refer to the *Agent Interaction SDK 7.5 Java Developer's Guide* on the Genesys Documentation Library DVD.

The JavaScript API Content

Because you should minimize the JavaScript code running on the browser side, the information provided by the JavaScript API is limited to the following items:

- Some events
- `userName`
- `idInteraction`

You can retrieve all other information, such as attached data, history, or contacts, from the AIL. The code examples in Chapter 2 show how to access the AIL.

The custom servlet or JSP can retrieve the interaction details from the AIL, and it can consolidate the response to the browser with some information coming from a third-party system.

You can create your own servlet, or JSP, and customize it to fit your needs. Just as you can call the AIL, you can call another system, through RMI or any other invocation method. You could, for example, access your own database through JDBC (Java DataBase Connectivity).

Limitations and Restrictions

Please note the following limitations and restrictions as you customize Genesys Agent Desktop:

1. Only one AIL runs in the Genesys Desktop server; therefore, your custom code shares the same instance as the Genesys Agent Desktop GUI Layer, and has access to the same objects.
2. Although you can use DHTML custom code to create a pop-up window (as outlined in "Creating a Pop-Up Window" on page 33), you cannot add any code for graphics objects in the Genesys Desktop frame.
3. All the custom code variable and function identifiers must be prefixed by `custom_`, for example, `custom_myvar`.

4. The DHTML code loaded into the `Tab` frame cannot refer to the `Parent` frame. All links must be relative to the current frame. (References to the `window.parent` frame are not supported.)
5. Some information, such as the current contents of the e-mail, current notepad, and so on, exist only within the browser and are accessible from the AIL only when the agent clicks the `Save` button.
6. Custom servlets must be deployed within the Genesys agent-facing interface. You can create your own servlets and place them in the `webapps/WEB-INF/classes/` directory.

Customization Examples

This section presents five code examples that illustrate possible ways to customize the Genesys Desktop 7.6 graphical user interface (GUI). Genesys recommends that you carefully examine these examples before you begin customizing Genesys Desktop. See *Genesys Desktop 7.6 XML Tag Reference* for detailed descriptions of the tags used in these examples.

This section contains the following sub-sections:

Shipped Code Examples

The following examples are shipped on Genesys Desktop DVD:

- `sample1—desktop`: Create a pop-up window at desktop initialization, as described in “Creating a Pop-Up Window” on page 33.
- `sample2—menu-bar`: Create an additional menu with menu item described in “Creating a Menu” on page 35.
- `sample3—button-bar`: Create an additional button to the `Interaction` toolbar, as described in “Creating a Button” on page 40.
- `sample4—customer-records`: Create a new tab in the customer record area, as described in “Creating a Tab Frame” on page 42.

- `sample5—toolbar`: Create an additional button in the `Desktop` toolbar, as described in "Creating a Toolbar Item" on page 36.
- `sample6—interaction-information`: This example is not described in this chapter. It allows you to label an interaction in ways to help routing and categorizing. Comments are provided in the shipped example itself.
- `sample7—contact-ext`: This sample is not described in this chapter. It enables you to replace the standard contact panel in an interaction.
- `Sample8—shortcut`: This sample enables you to create a custom shortcut, as described in "Creating a Shortcut" on page 47.
- `Sample9—central panel`: This sample enables you to display custom panels in the area where interactions and contact management panels are displayed by Genesys Desktop, as described in "Using Central Panel" on page 48.
- `Sample10—client notification`: This sample enables you to publish information, such as an event, from the server side to the web browser, using the Genesys Desktop `realtime` channel, as described in "Using Client Notification" on page 50.
- `sample11—scheduling client notification`: This sample is not described in this chapter. It enables you to schedule events that are sent to the agent. It employs features such as `custom event`, `shortcut`, and `central panel area`.
- `sample12—external knowledgebase`: This sample shows you how to insert text from an external resource into an interaction, such as outbound media, chat, SMS session, and so on. External resources are displayed in the tabs that correspond to `<resources>` and `<interaction-information>`.
- `sample13—custom area`: This sample shows how to add custom panels to the desktop area.
- `sample14—optimized size`: This sample shows you how to optimize the size of the desktop. Two sizes are demonstrated:
  - `800 x 150`—When nothing is displayed in central page.
  - `800 x 600`—For all other cases.

The examples in this chapter show how to customize various elements of the Desktop Frame, the main frame of the Genesys Desktop application.

Agent Desktop Frame

The `Desktop` frame (shown in Figure 6) is the main frame of the Genesys Desktop application. It contains the following items (see the *Genesys Desktop Agent Help* for a detailed explanation of each of these items):

- Batch navigation bar
- Menu
- Toolbar area

- Status bar
- `Interaction` frame
- Hidden frame that handles the real-time channel

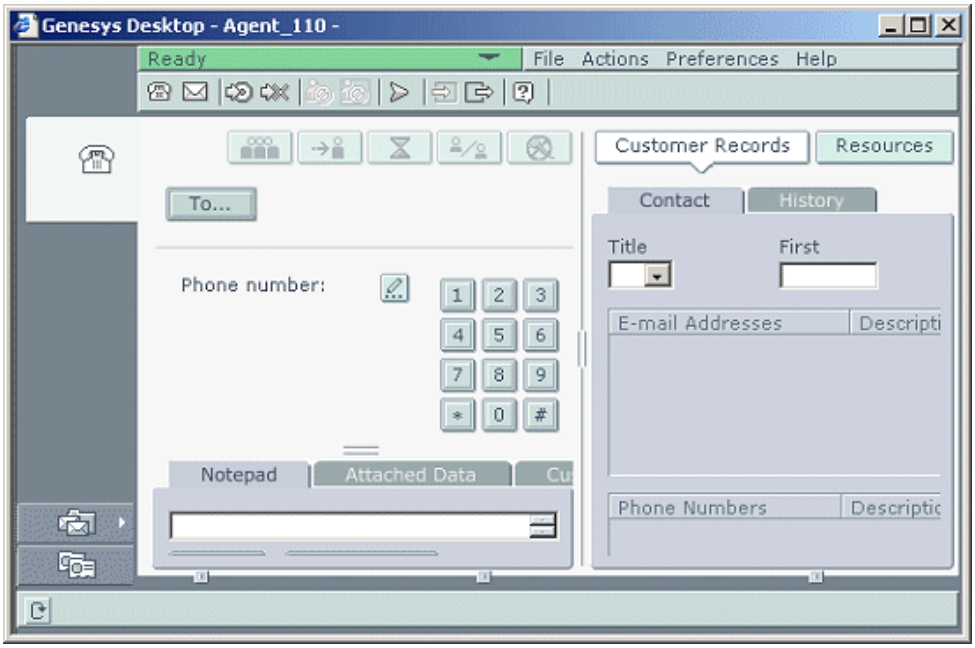


Figure 6: Desktop and Interaction Frames

Creating a Pop-Up Window

This example shows you how to create a pop-up window that appears when the Genesys Agent Desktop main frame loads.

Example Code

1. Create the JSP file, `custom/welcome.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html; charset=utf-8" %>
<html>
  <head>
    <title>Welcome</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <script language="javascript">
      window.opener.includeGDFunctionalities(window);

      function custom_load() {
        activateGDFunctionalities();
      }

      function custom_unload() {
        unactivateGDFunctionalities();
```

```
      }
    </script>
  </head>
  <body bgcolor="#ccccff" onLoad="custom_load()" onUnload="custom_unload()">
    <center><b><font size="6" color="#000099">
<%
    String isSupervisorParam = request.getParameter( "isSupervisor" );
    if ( (isSupervisorParam == null) || (!isSupervisorParam.equals("true")) ) {
%>
                Welcome to Agent <%= request.getParameter( "userName" ) %>
<%
    } else {
%>
                Welcome to Supervisor <%= request.getParameter( "userName" ) %>
<%
    }
%>
    </font></b></center>
    <hr>
  </body>
</html>
```

2. Create the `custom/custom.xml` customization file:

```
<gcn-resources>
   <desktop>
      <javascript-onload>
         <![CDATA[
         document.forms.welcomeForm.elements.userName.value =
                  userName;
            document.forms.welcomeForm.submit();
         ]]>
      </javascript-onload>
      <html-body>
         <![CDATA[
            <form name="welcomeForm" target="_blank"
                  method="get"
                  action="custom/welcome.jsp">
               <input type="hidden" name="userName"/>
            </form>
         ]]>
      </html-body>
   </desktop>
</gcn-resources>
```

3. Modify the `WEB-INF/web.xml` file to specify the customization file that you created in Step 2:

```
<servlet>
   <servlet-name>initUAD</servlet-name>
```

```
      .
      <init-param>
         <param-name>customFile</param-name>
         <param-value>/custom/custom.xml</param-value>
      </init-param>
   .
</servlet>
```

4. Start Genesys Desktop to see the pop-up window.

Creating a Menu

The menu bar is embedded in the `Desktop` frame. You can add menus (with submenus) between the `Preferences` and `Help` menus (as shown in Figure 7). Item 1 indicates a menu and item 2 indicates a submenu.

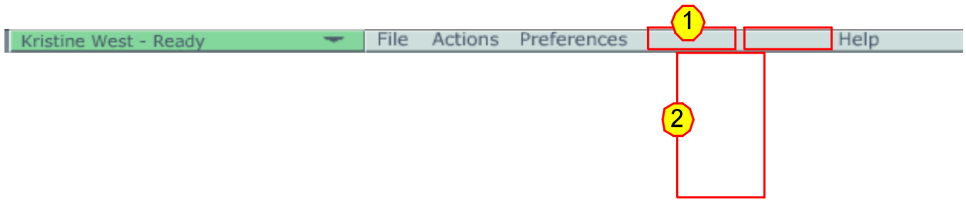


Figure 7: Default Menu Bar

There are two steps to adding custom menus:

1. Create a `Menu` tab.
2. Create a submenu.

Note: Do not put more than four items in each submenu. Some browsers might not correctly handle pop-up overlapping, or the `Interaction` frame might hide long submenus.

Example Code

This example shows you how to create a menu that displays an alert message on the agent desktop when an item is selected from this custom menu.

1. Create the customization file `custom/custom.xml`:

```
<gcn-resources>
  <desktop>
     <dictionary-class>custom</dictionary-class >
     <menu-bar>
        <display>true</display>
        <menus>
           <menu name="custom">
              <dictionary-key>menu.custom</dictionary-key>
```

```
                <menu-items>
                   <menu-item name="welcome">
                      <dictionary-key>menu.custom.welcome
                      </dictionary-key>
                      <javascript-onselect>
                         <![CDATA[ alert( "Welcome" ); ]]>
                      </javascript-onselect>
                   </menu-item>
                </menu-items>
             </menu>
          </menus>
       </menu-bar>
    </desktop>
</gcn-resources>
```

2. Modify the `WEB-INF/web.xml` file to specify the customization file that you created in Step 1:

```
<servlet>
   <servlet-name>initUAD</servlet-name>
   .
   <init-param>
      <param-name>customFile</param-name>
      <param-value>/custom/custom.xml</param-value>
   </init-param>
   .
</servlet>
```

3. Create a dictionary file, `WEB-INF/classes/custom.properties`, which contains the following menu labels:

```
menu.custom=Custom menu
menu.custom.welcome=Welcome
```

> **Note:** You can use space characters as part of the value of a key.

4. Start Genesys Desktop to see the custom menu.
5. Click the menu and choose the `Welcome` item. You should see the alert that welcomes you by your user name.

Creating a Toolbar Item

You can extend the Desktop toolbar by adding buttons that appear to the left of the `Help` button (as shown in Figure 8).

Figure 8: Creating a Toolbar Item

Example Code

This example shows you how to create a button that toggles agent status from `Not Ready` to `Ready`:

1. Create a JSP page, `custom/readyplace.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html; charset=utf-8"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script language="javascript">
      window.parent.includeGDFunctionalities(window);
      function custom_load() {
        activateGDFunctionalities();
      }

      function custom_unload() {
        unactivateGDFunctionalities();
      }
    </script>
  </head>
  <body height="0" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
   onLoad="custom_load()" onUnload="custom_unload()">
    <script language="javascript">
<%
    com.genesyslab.ail.AilFactory factory =
                                        com.genesyslab.ail.AilLoader.getAilFactory();
    com.genesyslab.ail.Person person = (com.genesyslab.ail.Person)
           factory.getPerson( (String)request.getParameter( "userName" ) );

    if (person instanceof com.genesyslab.ail.Agent) {
        com.genesyslab.ail.Agent agent = (com.genesyslab.ail.Agent)person;
        com.genesyslab.ail.Place place = agent.getPlace();

        java.util.Map readyDnExceptions = place.ready();
        java.util.Map readyMediaExceptions = place.readyMultimedia(null,"","");

        String readyDnIdsFailed = "";
        java.util.Iterator it = readyDnExceptions.entrySet().iterator();
        java.util.Map.Entry entry = null;
        java.lang.Exception exception = null;
        while(it.hasNext()) {
             entry = (java.util.Map.Entry) it.next();
             exception =  (java.lang.Exception) entry.getValue();
             if (exception != null) {
                String dnId = (String) entry.getKey();
                if (readyDnIdsFailed.equals(""))
                    readyDnIdsFailed += ", ";
                readyDnIdsFailed += dnId;
             }
```

```
        }

        String readyMediaNamesFailed = "";
        it = readyMediaExceptions.entrySet().iterator();
        while(it.hasNext()) {
             entry = (java.util.Map.Entry) it.next();
             exception =  (java.lang.Exception) entry.getValue();
             if (exception != null) {
                String mediaName = (String) entry.getKey();
                if (readyMediaNamesFailed.equals(""))
                    readyMediaNamesFailed += ", ";
                readyMediaNamesFailed += mediaName;
             }
        }

        String warningMessage = "";
        if (!readyDnIdsFailed.equals(""))
            warningMessage = "Ready failed on following dn(s): " + readyDnIdsFailed;
        if (!readyMediaNamesFailed.equals("")) {
            if (warningMessage.equals(""))
                warningMessage = "Ready failed on following media(s): ";
            else
                warningMessage += ", and on following media(s): ";
            warningMessage += readyMediaNamesFailed;
        }
        if (!warningMessage.equals("")) {
%>
      showWarningMessage("<%= warningMessage %>");
<%
        }
    }
%>
    </script>
  </body>
</html>
```

2. Create a custom dictionary file, `WEB-INF/classes/custom.properties`:

```
toolbar.ready=Ready
```

> **Note:** You can use space characters in the value of a key.

3. Create the customization file `custom/custom.xml`:

```
<gcn-resources>
    <desktop>
        <dictionary-class>custom</dictionary-class>
        <tool-bar>
            <display>true</display>
            <tool-bar-items>
```

```
                <tool-bar-item name="readyTool-bar-item">
                    <image-up-name>custom/btnready_up.gif</image-up-name>
                    <image-disable-name>custom/btnready_disable.gif</image-disable-name>
                    <image-down-name>custom/btnready_down.gif</image-down-name>
                    <image-focus-name>custom/btnready_focus.gif</image-focus-name>
                    <tooltip-dictionary-key>toolbar.ready</tooltip-dictionary-key>
                    <javascript-onselect>
                        <![CDATA[
                            document.forms.readyPlaceForm.elements.userName.value =
                                                                    userName;
                            document.forms.readyPlaceForm.submit();
                        ]]>
                    </javascript-onselect>
                    <html-body>
                        <![CDATA[
                            <form name="readyPlaceForm" method="get"
                                                action="custom/readyplace.jsp">
                                <input type="hidden" name="userName">
                            </form>
                        ]]>
                    </html-body>
                </tool-bar-item>
            </tool-bar-items>
        </tool-bar>
    </desktop>
</gcn-resources>
```

4. Create three bitmap files, `btnready_up.gif` for the button in the `Ready` state, `btnready_down.gif` for the `selected` button, `btnready_disable.gif` for the button in the `Disabled` state, and `btnready_focus.gif` for the button in the `Focused` state.
5. Place the buttons in the `custom/` directory.
6. Modify the `WEB-INF/web.xml` file to include the customization file that you created in Step 3:

```
<servlet>
   <servlet-name>initUAD</servlet-name>
   .
   <init-param>
      <param-name>customFile</param-name>
      <param-value>/custom/custom.xml</param-value>
   </init-param>
   .
</servlet>
```

7. Start up Genesys Desktop to see your custom button on the `Desktop` toolbar.

Creating a Button

You can extend the `Button` toolbar, by adding buttons that appear to the left of the regular buttons (as shown in Figure 9).

Figure 9: Default Button Toolbar

Example Code

This example shows you how to create a button that releases the established call:

1. Create the JSP page, `custom/releaseInteraction.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html; charset=utf-8"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script language="javascript">
      window.parent.includeGDFunctionalities(window);

      function custom_load() {
        activateGDFunctionalities();
      }

      function custom_unload() {
        unactivateGDFunctionalities();
      }
    </script>
  </head>
  <body height="0" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
   onLoad="custom_load()" onUnload="custom_unload()">
    <script language="javascript">
<%
    com.genesyslab.ail.AilFactory factory =
                                      com.genesyslab.ail.AilLoader.getAilFactory();
    com.genesyslab.ail.Interaction interaction =
           factory.getInteraction( (String)request.getParameter( "idInteraction" ) );

    try {
        if( com.genesyslab.ail.Interaction.Type.PHONE_CALL.equals(
                                                        interaction.getType() ) ) {
            interaction.releaseCall( null );
        }
    } catch( com.genesyslab.ail.exception.RequestFailedException
                                                         requestFailedException ) {
%>
```

```
    showWarningMessage("Release interaction failed!");

<%
    }
%>
    </script>
  </body>
</html>
```

2. Create a custom dictionary file, `WEB-INF/classes/custom.properties`:

```
interaction.release=Release
```

Note: You can use space characters in the value of a key.

3. Create the customization file `custom/custom.xml`:

```
<gcn-resources>
   <desktop>
      <interaction-action>
         <buttons>
            <button name="releaseButton">
               <image-up-name>
                  custom/btnrelease_up.gif
               </image-up-name>
               <image-disable-name>
                  custom/btnrelease_disable.gif
               </image-disable-name>
               <image-focus-name>
                  custom/btnready_focus.gif
               </image-focus-name>
               <tooltip-dictionary-key>
                  interaction.release
               </tooltip-dictionary-key>
               <javascript-oninteractionstatuschange>
                  <![CDATA[
                     if( interactionStatus == STATUS_TALKING )
                        enableButton( "releaseButton" );
                     else
                        disableButton( "releaseButton" );
                  ]]>
               </javascript-oninteractionstatuschange>
               <javascript-onload>
                  <![CDATA[
                     if( interactionStatus == STATUS_TALKING )
                        enableButton( "releaseButton" );
                     else
                        disableButton( "releaseButton" );
                     ]]>
               </javascript-onload>
```

```
            <javascript-onselect>
               <![CDATA[
document.forms.releaseForm.elements.idInteraction.value =
   idInteraction;
document.forms.releaseForm.submit();
               ]]>
            </javascript-onselect>
            <html-body>
               <![CDATA[
                  <form name="releaseForm" method="get"
                     action="custom/releaseInteraction.jsp">
                     <input type="hidden"
                        name="idInteraction">
                  </form>
               ]]>
            </html-body>
         </button>
      </buttons>
   </interaction-action>
</desktop>
</gcn-resources>
```

4. Create three bitmap files, `btnrelease_up.gif` for the button in the `Ready` state, `btnrelease_disable.gif` for the button in the `Disabled` state, and `btnrelease_focus.gif` for the button in the `Disabled` state.
5. Place the bitmap files in the `custom/` directory.
6. Modify the `WEB-INF/web.xml` file to include the customization file that you created in Step 3:

```
<servlet>
   <servlet-name>initUAD</servlet-name>
   .
   <init-param>
      <param-name>customFile</param-name>
      <param-value>/custom/custom.xml</param-value>
   </init-param>
   .
</servlet>
```

7. Start Genesys Desktop to see your custom button in the `Button` toolbar.
8. Activate a call.
9. Click the custom button to release the call.

Creating a Tab Frame

You can add several tab frames to the `Resources`, `Interaction Information`, and `Customer Records` frames of the Desktop Frame. Each tab can contain one

frame, which can display a DHTML page coming from any URL (as shown in Figure 10). The following frames are available:

- `Customer Records` frame—Use for contact information tabs.
- `Interaction Information` frame—Use for information about the interaction.
- `Resources` frame—Use for all other types of information.

This customization adds a tab in the `Customer-Records` frame, as follows:

- If the interaction type is `INTERACTION_VOICE`, the `Added` tab is automatically selected.
- If the interaction type is `INTERACTION_CHAT`, the `History` tab is automatically selected.
- For any other interaction type, the `Contact` tab is automatically selected.

When you click a tab, the frame is loaded and displays some information about the contact and the interaction.

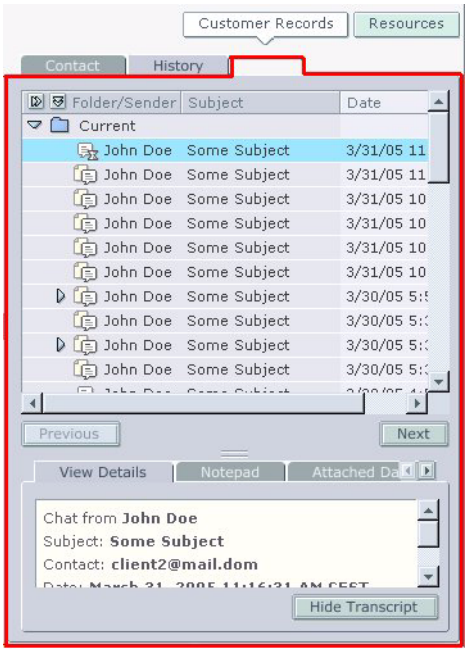


Figure 10: Example of a Tab in the Customer Records Frame

Example Code

This example shows you how to display a web page inside a custom tab.

1. Create a DHTML page, `custom/additionalInformation.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html; charset=utf-8" %>
<%
    com.genesyslab.ail.AilFactory factory =
   com.genesyslab.ail.AilLoader.getAilFactory();
```

```
    com.genesyslab.ail.Interaction interaction = factory.getInteraction(
   (String)request.getParameter( "idInteraction" ) );
    com.genesyslab.ail.ContactManager contactManager = factory.getContactManager();
    com.genesyslab.ail.Contact contact = factory.getContactManager().getContact(
   (String)request.getParameter( "idContact" ) ,null);
%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script language="javascript">
      window.parent.includeGDFunctionalities(window);

      function custom_load() {
        activateGDFunctionalities();
      }

      function custom_unload() {
        unactivateGDFunctionalities();
      }
    </script>
  </head>
  <body style="font-size: 10pt;" bgcolor="#ffcccc" onLoad="custom_load()"
   onUnload="custom_unload()">
<% if( contact != null ) { %>
        Contact: <b><%=contact.getTitle()%> <%=contact.getFirstName()%>
   <%=contact.getLastName()%></b><br>
        Primary Phone Number: <b><%=contact.getPrimaryPhoneNumber()%></b><br>
        Primary E-mail Address: <b><%=contact.getPrimaryEmailAddress()%></b><br>
<% } else { %>
        <center><b>Contact Information Not Available</b></center><br>
<% } %>
        <hr>
<% if( interaction != null ) { %>
        Date of Creation: <b><%=interaction.getDateCreated()%></b><br>
        Agent: <b><%=interaction.getUserName()%></b><br>
        Subject: <b><%=interaction.getSubject()%></b><br>
    <% if( interaction.getDn() != null ) { %>
        Dn: <b><%=interaction.getDn().getId()%></b><br>
    <% } else { %>
        Dn: <b>No Dn Information Available</b><br>
    <% } %>
<% } else { %>
    <center><b>Interaction Information Not Available</b></center><br>
<% } %>
  </body>
</html>
```

2. Create a custom dictionary file, `WEB-INF/classes/custom.properties`:

```
AdditionalInformation=Additional Information
```

Note: You can use space characters in the value of a key.

3. Create the customization file, `custom/custom.xml`:

```
<gcn-resources>
   <desktop>
      <dictionary-class>custom</dictionary-class>
      <customer-records>
         <javascript-onload>
            <![CDATA[
               switch(interactionType) {
                  case INTERACTION_CHAT:
                     selectTabByName(TAB_CUSTOMER_RECORDS_HISTORY);
                     break;
                  case INTERACTION_VOICE:
                     selectTabByName("additionalInformation");
                     break;
               }
            ]]>
         </javascript-onload>
         <tabs>
            <tab name="additionalInformation">
               <dictionary-key>additionalInformation</dictionary-key>
               <javascript-onselect>
                  <![CDATA[
document.forms.additionalInformationForm.target = getDetailFrameName();
document.forms.additionalInformationForm.elements.idInteraction.
value = idInteraction;

   document.forms.additionalInformationForm.elements.idContact.value = idContact;
                           document.forms.additionalInformationForm.submit();
                  ]]>
               </javascript-onselect>
               <html-body>
                  <![CDATA[
                     <form name="additionalInformationForm"
                        action="custom\additionalInformation.jsp">
                        <input type="hidden" name="idInteraction">
                        <input type="hidden" name="idContact">
                     </form>
                  ]]>
               </html-body>
            </tab>
         </tabs>
      </customer-records>
   </desktop>
</gcn-resources>
```

> **Note:** To display a web page from another site (for co-browsing), enter the URL as the value for the `action` attribute of the `<form>` tag that is nested inside the `<html-body>` tag.

4. Modify the `WEB-INF/web.xml` file to specify the customization file that you created in Step 3:

```
<servlet>
   <servlet-name>initUAD</servlet-name>
   .
   <init-param>
      <param-name>customFile</param-name>
      <param-value>/custom/custom.xml</param-value>
   </init-param>
   .
</servlet>
```

5. Restart Genesys Desktop.

Limitations for Tabs

In creating custom tabs, keep the following limitations in mind:

- Your custom code cannot reference the Genesys Agent Desktop parent frame, which is the `Customer Records` frame. All links must be relative to your custom web page. You cannot reference the following upper-level frames: `Customer Records`, `Interaction`, `Desktop`, nor any others at this level.
- The browser does not provide an automatic caching mechanism. This means that if another tab or another interaction is selected, the custom page unloads. You can implement manual caching using the `setCustomContext` and `getCustomContext` functions available in the `<tab>` tag of the `custom.xml` file.

Creating a Shortcut

You can create a custom shortcut that is configurable in the `Extended` section of the `Shortcuts` panel on the Genesys Desktop GUI (see Figure 11).

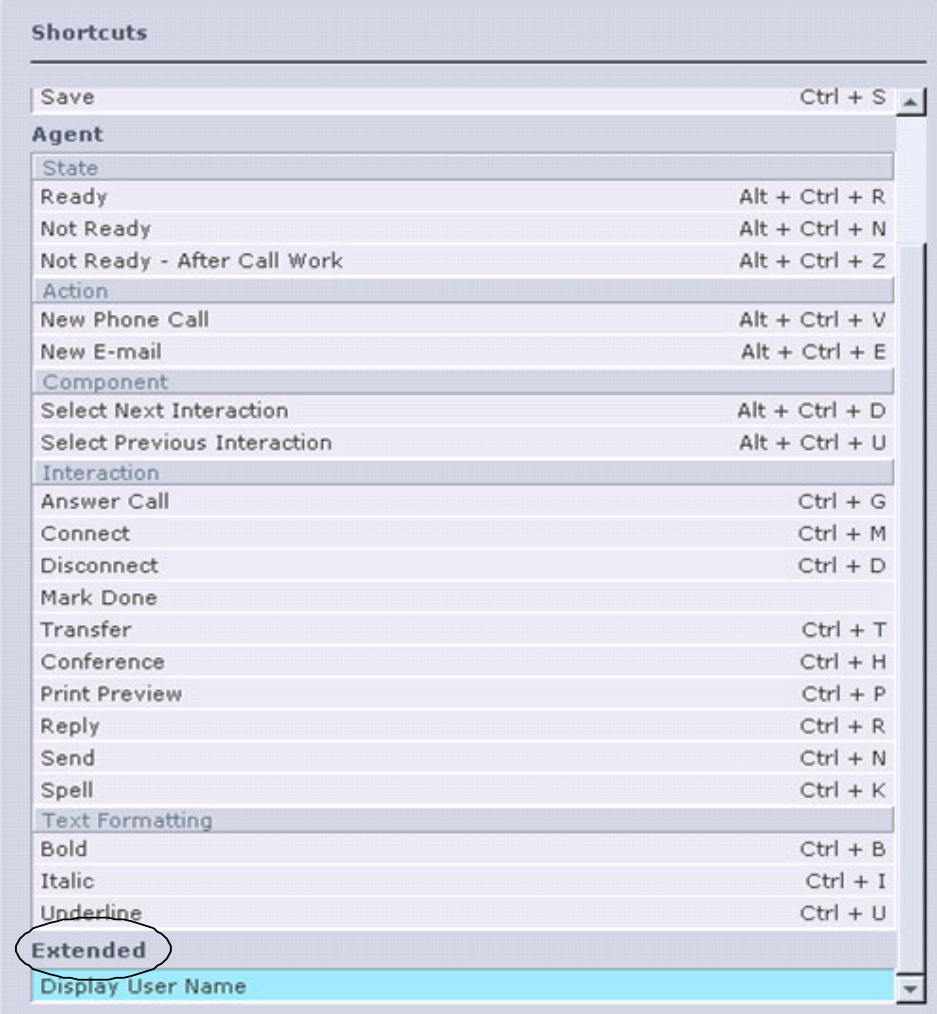


Figure 11: Genesys Desktop Shortcuts Panel Showing the Extended Section Where You Can Configure a Custom Shortcut

Example Code

This example shows you how to create a shortcut that displays the user name in an alert message on the agent desktop when the custom shortcut is applied.

1. Create the customization file `custom/custom.xml`:

```
<gcn-resources>
  <desktop>
    <shortcuts>
      <shortcut name="displayUserName">
```

```
            <text>Display User Name</text>
            <javascript-onexecute>
               <![CDATA[
                  alert(userName);
               ]]>
            </javascript-onexecute>
         </shortcut>
      </shortcuts>
   </desktop>
</gcn-resources>
```

2. Modify the `WEB-INF/web.xml` file to specify the customization file `custom/custom.xml`:

```
<servlet>
   <servlet-name>initUAD</servlet-name>
   .
   <init-param>
      <param-name>customFile</param-name>
      <param-value>/custom/custom.xml</param-value>
   </init-param>
   .
</servlet>
```

3. Start Genesys Desktop.
4. Open the Shortcuts pane.
5. Configure the custom shortcut and apply it. An alert should be displayed that shows your user name.

Using Central Panel

You can display custom panels in the area where interactions and contact management panels are displayed by Genesys Desktop. This area is referred to as the `Central Panel`.

Example Code

This example shows you how to display information in the `Central Panel` and then close it.

1. Create the JavaScript file `htmlcontrol.js` (refer to “htmlcontrol.js” on page 100)
2. Create the JSP page `custom/user-name.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html;
charset=utf-8" %>
<html>
```

```
  <head>
    <meta http-equiv="Content-Type" content="text/html;
     charset=utf-8">
    <script language="javascript" src="htmlcontrol.js"></script>
    <script language="javascript">
      window.parent.includeGDFunctionalities(window);

      function custom_load() {
        activateGDFunctionalities();
        addControl(custom_closeButton);
        focusOnControl(custom_closeButton);
      }

      function custom_unload() {
        unactivateGDFunctionalities();
      }
    </script>
  </head>
  <body bgcolor="#ccccff" onLoad="custom_load()"
onUnload="custom_unload()">
    <center><b>
      <font size="6" color="#000099">
        User name:<%=request.getParameter( "userName" )%>
      </font>
    </b></center>
    <hr>
    <div id="closeDiv"></div>
    <script language="javascript">
      var custom_closeButton = custom_createHTMLButtonImage
     ("closeDiv", "Close",
     ["CloseBtn.gif","CloseBtng.gif","CloseBtnd.gif"], 25,
     "parent.closeCentralPanel('DISPLAY_USERNAME')");
    </script>
  </body>
</html>
```

3. Create the customization file `custom/custom.xml`:

```
<gcn-resources>
   <desktop>
     <html-body>
       <![CDATA[
          <form name="displayUserNameForm" method="POST"
             action="custom/user-name.jsp">
             <input type="hidden" name="userName"/>
          </form>
       ]]>
     </html-body>
     <menu-bar>
       <display>true</display>
```

```
            <menus>
               <menu name="custom">
                  <text>Custom</text>
                  <menu-items>
                     <menu-item name="displayUserName">
                        <text>Display User Name</text>
                        <javascript-onselect>
                           <![CDATA[
                              document.forms.displayUserNameForm.
elements.userName.value = userName;
                              openCentralPanel("DISPLAY_USERNAME",
"displayUserNameForm");
                           ]]>
                        </javascript-onselect>
                     </menu-item>
                  </menu-items>
               </menu>
            </menus>
      </menu-bar>
   </desktop>
</gcn-resources>
```

4. Modify the `WEB-INF/web.xml` file to include the customization file `custom/custom.xml`:

```
<servlet>
<servlet-name>initUAD</servlet-name>
.
<init-param>
<param-name>customFile</param-name>
<param-value>/custom/custom.xml</param-value>
</init-param>
.
</servlet>
```

5. Start Genesys Desktop to verify that your custom button is on the `Button` toolbar.
6. Click the menu and choose `Display User Name`. Agent information should be displayed in the central panel.

Using Client Notification

You can publish information, such as an event, from the server side to the web browser using the Genesys Desktop `realtime channel`.

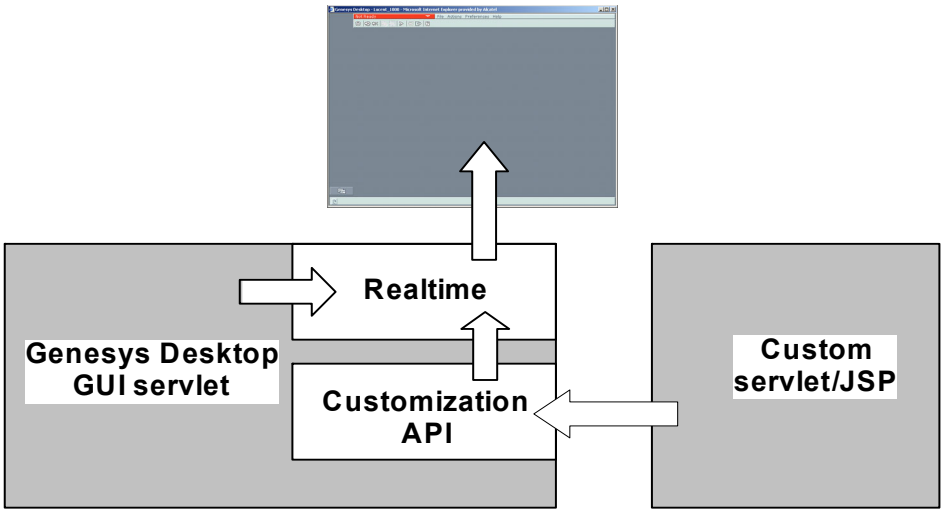


Figure 12: Genesys Desktop GUI Layer and Custom Code Employing the Realtime Channel

To publish an event from your JSP or servlet, you should employ an instance of the `com.genesyslab.uadthin.customization.realtime` interface that is provided by the `customEventPublisher` attribute of the `javax.servlet.http.HttpSession` servlet session. The instance is added at the `login` step. Use the following `publish` method:

```
public void publish(String data[]);
```

The `data` parameter is sent to the browser by the `realtime` channel. You can capture it in a `javascript-oncustomevent` tag using the `customEvent` JavaScript variable.

Note: This interface is included in the following Genesys Desktop customization `.jar` file:

`WEB-INF/lib/gdesktop-customization-api.jar`

You can use this in your development environment to compile your java file.

Example Code

This customization example adds a menu in the menu-bar that enables you to select the item `Start Client Notification` to send a request to the Genesys Desktop Server to publish two custom events. The browser displays the event attributes in an alert message when they are received.

1. Create the DHTML page `custom/initialize.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html;
charset=utf-8" %>
<%@ page import="java.util.Timer" %>
```

```
<%@ page import="java.util.TimerTask" %>
<%@ page
import="com.genesyslab.uadthin.customization.realtime.CustomEventPu
blisher" %>
<%!
    CustomEventPublisher customEventPublisher;
    Timer timer = new Timer(true);
%>
<%
    customEventPublisher =
(CustomEventPublisher)session.getAttribute("customEventPublisher");
    if (customEventPublisher != null) {
        timer.schedule(new TimerTask() {
            public void run() {
                customEventPublisher.publish(new String[]{"First
message"});
            }
        },1000);
        timer.schedule(new TimerTask() {
            public void run() {
                customEventPublisher.publish(new String[]{"Second
message"});
            }
        },6000);
    }
%>
<html>
  <head>
   <script language="javascript">
      window.parent.includeGDFunctionalities(window);

      function custom_load() {
        activateGDFunctionalities();
      }

      function custom_unload() {
        unactivateGDFunctionalities();
      }
    </script>
  </head>
  <body onLoad="custom_load()" onUnload="custom_unload()">
  </body>
</html>
```

2. Create the customization file `custom/custom.xml`:

```
<gcn-resources>
  <desktop>
    <html-body>
      <![CDATA[
```

```
            <form name="initializeForm" method="POST"
target="hiddenCustomFrame"
               action="custom/initialize.jsp">
            </form>
               <iframe name="hiddenCustomFrame"
style="visibility:hidden"></iframe>
         ]]>
      </html-body>
      <menu-bar>
         <display>true</display>
         <menus>
            <menu name="custom">
               <text>Custom</text>
               <menu-items>
                  <menu-item name="startClientNotification">
                     <text>Start Client Notification</text>
                     <javascript-onselect>
                        <![CDATA[
                           document.forms.initializeForm.submit();
                        ]]>
                     </javascript-onselect>
                  </menu-item>
               </menu-items>
            </menu>
         </menus>
      </menu-bar>
      <javascript-oncustomevent>
         <![CDATA[
            alert("Client notification: " + customEvent[0]);
         ]]>
      </javascript-oncustomevent>
   </desktop>
</gcn-resources>
```

3. Modify the file `WEB-INF/web.xml` that you created above to specify the customization file:

```
<servlet>
<servlet-name>initUAD</servlet-name>
.
<init-param>
<param-name>customFile</param-name>
<param-value>/custom/custom.xml</param-value>
</init-param>
.
</servlet>
```

4. Restart Genesys Desktop.

5. Click the `Custom` menu and select `Start Client Notification`. Two alert message should be displayed; the first after 1 second, and the second after 6 seconds.

Inserting Text from an External Knowledge Base

You can insert information into a chat, instant message, or e-mail interaction view from an external knowledge base by using the external knowledgebase customization. By default, the `append` action is disabled.

The following JavaScript functions are available for the `<customer-records>`, `<interaction-information>`, and `<resources>` tags:

- `appendTextInInteraction(text)`—Appends information in an interaction using the plain text format.
- `appendHTMLInInteraction(html)`—Appends information in an interaction using the HTML format. If the media does not support the HTML format, it is converted to plain text before the append action is applied.
- `isPossibleAppendInformation`—return true if it is possible to append information

The `<javascript-onappendinformationstatechange>` tag is available in the `<customer-records>`, `<interaction-information>` and `<resources>` tags.

The `<javascript-onappendinformationstatechange>` tag is called when the state of the interaction changes to a state in which it becomes possible to append information to the interaction; for example the interaction state changes from RINGING to TALKING. It is also called when the state of the interaction changes to a state in which it becomes impossible to append information; for example, the interaction state changes from TALKING to IDLE.

The `<javascript-onappendinformationstatechange>` tag is not mandatory to implement a customization.

1. Create the JavaScript file `htmlcontrol.js` (refer to “htmlcontrol.js” on page 100).
2. Create the DHTML page `custom/external-knowledgebase.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html; charset=utf-8" %>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <link rel="stylesheet" type="text/css" href="style.css"/>
    <script language="javascript" src="htmlcontrol.js"></script>
    <script language="javascript">
      window.parent.includeGDFunctionalities(window);

      function custom_load(){
        activateGDFunctionalities();
```

```
      addControl(custom_tableControl);
      addControl(custom_insertButton);

      focusOnControl(custom_tableControl);
    }

    function custom_unload(){
      unactivateGDFunctionalities();
    }

    function custom_changeInsertStatus(status){
      if(status)
        custom_insertButton.enable();
      else
        custom_insertButton.disable();
     }

     function custom_insertText(){
       var text =
custom_tableControl.getRowValue(custom_tableControl.getRowSelected(), 1);
        if(custom_tableControl.getRowSelected() == 0)
          window.parent.appendTextInInteraction(text);
        else
          window.parent.appendHTMLInInteraction(text);
      }
    </script>
 </head>
 <body class="fontName backgroundPanel" leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0" bgcolor="#FF0000" onLoad="custom_load()"
onUnload="custom_unload()">
 <table class=width="100%" border="0" cellpadding="0" cellspacing="0">
   <tr valign="top">
     <td class="backgroundPanel" align="left">
       <table class="editablePanelText" id="knowledgeTable" name="knowledgeTable"
width="100%" border="0" cellspacing="6" cellpadding="6">
         <tr id="row0" height="40px" class="tableBackgroundItem">
              <td width="30%" id ="row0_0">Some Plain Text</td>
            <td width="70%" id="row0_1">
                 This is some plain text only.
              </td>
         </tr>
         <tr id="row1" height="40px" class="tableBackgroundItem">
            <td width="30%" id="row1_0">Some HTML Text</td>
            <td width="70%" id="row1_1"><B>This is some bold
text</B><I>italic</I><FONT COLOR="GREEN"><u>green & underlined</U></FONT><A
HREF="javascript:alert('help')">help</A>
            </td>
          </tr>
       </table>
     </td>
   </tr>
```

```
    <tr align="center">
     <td>
       <div id="insertDiv"></div>
     </td>
    </tr>
  </table>
  <script type="text/javascript">
    var custom_tableControl = custom_createHTMLTableControl("knowledgeTable");
    var custom_insertButton = custom_createHTMLButtonImage ("insertDiv", "Insert",
["btninserts.gif","btninsertg.gif","btninsertd.gif"], 25, "custom_insertText()");

    if(!window.parent.isPossibleAppendInformation()){
      custom_insertButton.disable();
    }

    function custom_createHTMLTableControl(id){
      function HTMLTable(id){
        this.id = id;
        this.elm = window.document.getElementById(id);
        this.focused = false;
        this.rowSelected = 0;
        this.rows = 0;

        this.activate();
      }
      HTMLTable.prototype = {
        activate:function() {
          var rows =
this.elm.getElementsByTagName('tbody')[0].getElementsByTagName('tr');
          this.rows = rows.length;
          this.elm.onclick = createCallback(this,this.focus,[]);

          for (i = 0; i < rows.length; i++) {
            rows[i].onclick = createCallback(this,this.selectRow,[i]);
          }
        },

        focus:function(){
          if(!this.focused){
            this.focused = true;
            this.elm.focus();
            if(this.rows > 0)
              this.selectRow(this.rowSelected);
            focusOnControl(this);
          }
        },

        unfocus:function(){
          this.focused = false;
          if(this.rows > 0)
            this.selectRow(this.rowSelected);
```

```
            unfocusOnControl(this);
          },

          isFocusable:function(){
             return true;
          },

          onArrowUp:function(){
            if(this.rowSelected > 0)
              this.selectRow(this.rowSelected - 1);
          },

          onArrowDown:function(){
            if(this.rowSelected < (this.rows-1))
              this.selectRow(this.rowSelected + 1);
          },

          getRowSelected:function(){
            return this.rowSelected;
          },

          getRowValue:function(row, cell){
            var element = document.getElementById("row"+row);
            if(element != null){
              var cells = element.getElementsByTagName('td');
               if(cells.length > 1)
                 return cells[cell].innerHTML;
             }
             return "";
          },

          selectRow:function(row){
            if(this.rowSelected != row){
              var element = document.getElementById("row"+this.rowSelected);
              element.className = "tableBackgroundItem";
            }
            this.rowSelected = row;
            element = document.getElementById("row"+this.rowSelected);
            if(this.focused)
              element.className = "tableBackgroundSelectedFocusedItem";
            else
              element.className = "tableBackgroundSelectedItem";
          }
        }
        return new HTMLTable(id);
      }
    </script>
  </body>
</html>
```

3. Create the JavaScript page `custom/emoticons.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html; charset=utf-8" %>
<html>
    <head>
        <title>External Knowledgebase</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <link rel="stylesheet" type="text/css" href="style.css"/>
        <script language="javascript" src="htmlcontrol.js"></script>
       <script language="javascript">
        window.parent.includeGDFunctionalities(window);
        </script>
    <script language="JavaScript">
          function custom_changeInsertStatus(status){
                if(status)
                   custom_insertButton.enable();
                else
                   custom_insertButton.disable();
           }

           function custom_insertText(){
          var text =
   custom_tableControl.getRowValue(custom_tableControl.getRowSelected(), 0);
                window.parent.appendHTMLInInteraction(text);
           }
    </script>
    </head>
    <body class="fontName backgroundPanel" leftmargin="0" topmargin="0" marginwidth="0"
   marginheight="0" onLoad="javascript:custom_load()"
   onUnload="javascript:custom_unload()">
   <table width="100%" border="0" cellpadding="0" cellspacing="0">
      <tr valign="top">
        <td class="backgroundPanel" align="left">
          <table class="editablePanelText" id="knowledgeTable" name="knowledgeTable"
   width="100%" border="0" cellspacing="6" cellpadding="6">
           <tr height="16" id="row0" class="tableBackgroundItem">
           <td width="20%" id ="row0_0">:-)</td>
               <td width="70%" id="row0_1">Happy</td>
           </tr>
           <tr height="16" id="row1" class="tableBackgroundItem">
           <td width="20%" id ="row1_0">:]-[</td>
               <td width="70%" id="row1_1">Angry</td>
           </tr>
           <tr height="16" id="row2" class="tableBackgroundItem">
           <td width="20%" id ="row2_0">:-(</td>
               <td width="70%" id="row2_1" >Sad</td>
           </tr>
          </table>
        </td>
      </tr>
      <tr><td height="5" colspan="2"><img src="empty.gif"></td></tr>
```

```
    <tr align="center">
     <td>
     <div id="insertDiv"></div>
     </td>
    </tr>
  </table>
  <script type="text/javascript">
    var custom_tableControl = custom_createHTMLTableControl("knowledgeTable");
    var custom_insertButton = custom_createHTMLButtonImage ("insertDiv", "Insert",
["btninserts.gif","btninsertg.gif","btninsertd.gif"], 25, "custom_insertText()");

   if(!window.parent.isPossibleAppendInformation()){
      custom_insertButton.disable();
    }

    function custom_load(){
     activateGDFunctionalities();

      addControl(custom_tableControl);
      addControl(custom_insertButton);

      focusOnControl(custom_tableControl);
    }

    function custom_unload(){
     unactivateGDFunctionalities();
    }

    function custom_createHTMLTableControl(id){
     function HTMLTable(id){
        this.id = id;
        this.elm = window.document.getElementById(id);
        this.focused = false;
        this.rowSelected = 0;
        this.rows = 0;

        this.activate();
  }
  HTMLTable.prototype = {

    activate:function() {
            var rows =
this.elm.getElementsByTagName('tbody')[0].getElementsByTagName('tr');
            this.rows = rows.length;
            this.elm.onclick = createCallback(this,this.focus,[]);

          for (i = 0; i < rows.length; i++) {
                  rows[i].onclick = createCallback(this,this.selectRow,[i]);
          }
       },
```

```
focus:function(){
  if(!this.focused){
    this.focused = true;
   this.elm.focus();
    if(this.rows > 0)
     this.selectRow(this.rowSelected);
    focusOnControl(this);
  }
},

unfocus:function(){
  this.focused = false;
  if(this.rows > 0)
     this.selectRow(this.rowSelected);
  unfocusOnControl(this);
},

isFocusable:function(){
   return true;
},

    onArrowUp:function(){
      if(this.rowSelected > 0)
         this.selectRow(this.rowSelected - 1);
    },

    onArrowDown:function(){
      if(this.rowSelected < (this.rows-1))
         this.selectRow(this.rowSelected + 1);
    },

    getRowSelected:function(){
      return this.rowSelected;
    },

    getRowValue:function(row, cell){
       var element = document.getElementById("row"+row);
       if(element != null){
      var cells = element.getElementsByTagName('td');
      if(cells.length > 1)
     return cells[cell].innerHTML;
       }
       return "";
    },

    selectRow:function(row){
      if(this.rowSelected != row){
       var element = document.getElementById("row"+this.rowSelected);
      element.className = "tableBackgroundItem";
    }
       this.rowSelected = row;
```

```
            element = document.getElementById("row"+this.rowSelected);
            if(this.focused)
              element.className = "tableBackgroundSelectedFocusedItem";
            else
              element.className = "tableBackgroundSelectedItem";
          }
     }
    return new HTMLTable(id);
      }
    </script>
    </body>
</html>
```

4. Create the customization file `custom/custom.xml`:

```
<gcn-resources>
    <desktop>
        <resources>
            <tabs>
                <tab name="externalKnowledgebase">
                    <text>External Knowledgebase</text>
                    <javascript-onselect>
                        <![CDATA[
                            document.forms.externalKnowledgebaseForm.target =
   getDetailFrameName();

   document.forms.externalKnowledgebaseForm.elements.idInteraction.value =
   idInteraction;

   document.forms.externalKnowledgebaseForm.elements.selectedId.value = selectedId;
                            document.forms.externalKnowledgebaseForm.submit();
                        ]]>
                    </javascript-onselect>
                    <javascript-onappendinformationstatechange>
                        <![CDATA[
                          var custom_disabled = false;
                          if (!isPossibleAppendInformation()){
                              custom_disabled = true;
                     }
          var custom_iframe = document.getElementById(getDetailFrameName());
          if ( custom_iframe.contentDocument ) {

   if(custom_iframe.contentDocument.custom_changeInsertStatus)

   custom_iframe.contentDocument.custom_changeInsertStatus(!custom_disabled);
                          } else if ( custom_iframe.contentWindow ) {
                              if(custom_iframe.contentWindow.custom_changeInsertStatus)

   custom_iframe.contentWindow.custom_changeInsertStatus(!custom_disabled);
```

```
                        }
                    ]]>
                </javascript-onappendinformationstatechange>
                <html-body>
                    <![CDATA[
                        <form name="externalKnowledgebaseForm"
target="externalKnowledgebaseFrame" action="custom/external-knowledgebase.jsp">
                            <input type="hidden" name="idInteraction">
                            <input type="hidden" name="selectedId">
                        </form>
                        <iframe name="externalKnowledgebaseFrame"
id="externalKnowledgebaseFrame" style="visibility:hidden" frameborder="no" width="0"
height="0" src="custom/empty.htm"></iframe>
                    ]]>
                </html-body>
            </tab>
        </tabs>
    </resources>
    <interaction-information>
        <tabs>
            <tab name="externalKnowledgebase">
                <text>Emoticons</text>
                <scrollbar>true</scrollbar>
                <javascript-onselect>
                    <![CDATA[
                        document.forms.externalKnowledgebaseEmoticonsForm.target =
getDetailFrameName();

document.forms.externalKnowledgebaseEmoticonsForm.elements.idInteraction.value =
idInteraction;
                        document.forms.externalKnowledgebaseEmoticonsForm.submit();
                    ]]>
                </javascript-onselect>
                <javascript-onappendinformationstatechange>
                    <![CDATA[
                      var custom_disabled = false;
                      if (!isPossibleAppendInformation()){
                          custom_disabled = true;
                 }
       var custom_iframe = document.getElementById(getDetailFrameName());
       if ( custom_iframe.contentDocument ) {

if(custom_iframe.contentDocument.custom_changeInsertStatus)

custom_iframe.contentDocument.custom_changeInsertStatus(!custom_disabled);
                      } else if ( custom_iframe.contentWindow ) {
                          if(custom_iframe.contentWindow.custom_changeInsertStatus)

custom_iframe.contentWindow.custom_changeInsertStatus(!custom_disabled);
                      }
                    ]]>
```

```
                    </javascript-onappendinformationstatechange>
                    <html-body>
                        <![CDATA[
                            <form name="externalKnowledgebaseEmoticonsForm"
   target="externalKnowledgebaseEmoticonsFrame" action="custom/emoticons.jsp">
                                <input type="hidden" name="idInteraction">
                            </form>
                            <iframe name="externalKnowledgebaseEmoticonsFrame"
   style="visibility:hidden" frameborder="no" width="0" height="0"
   src="custom/empty.htm"></iframe>
                        ]]>
                    </html-body>
                </tab>
            </tabs>
        </interaction-information>
    </desktop>
</gcn-resources>
```

Using Custom Areas in the Agent Desktop Interface

Genesys Desktop customization enables you to expose and use custom areas within the Agent Desktop interface. Three different JavaScript functions are provided to enable you to access three different areas on the Agent Desktop, the top, the right, and the bottom (see Figure 13).

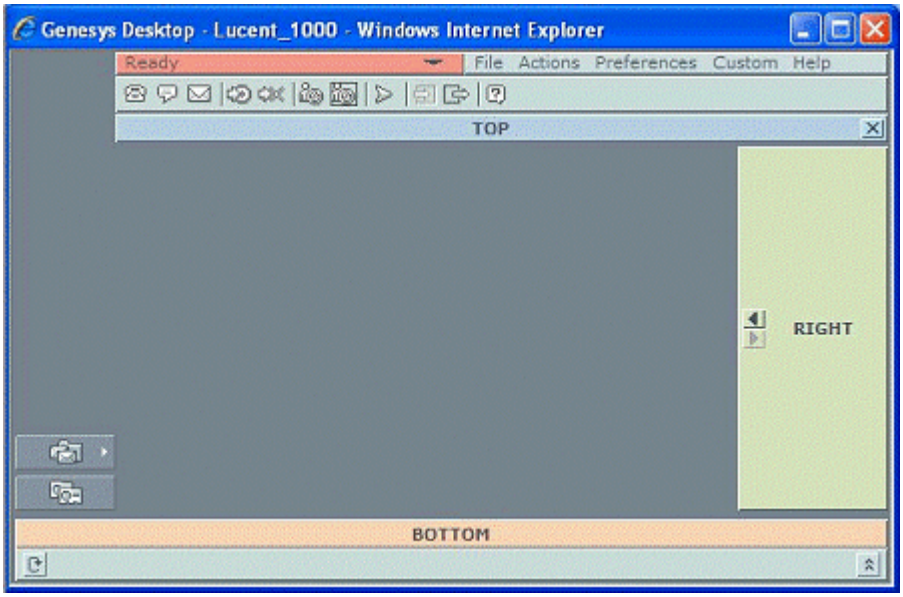


Figure 13: The three custom areas of the Genesys Desktop Agent interface where you can display custom content.

Each of these custom areas has a specific set of features. The top area has a close box that enables the agent to close the area (see Figure 13).

The right custom area has two buttons that enable agents to expand and contract the area (see Figure 13).

The bottom custom area has checked menu option that enables agents to show and hide the area (see Figure 13).

This example shows you how to expose and use custom areas within the Genesys Desktop interface. You do not have to create all three custom areas if you want to use one or two areas, only.

The following functions and constants are provided in the `<desktop>` tag to enable you to display custom areas in the Agent Desktop.

- Constants
  - `CUSTOM_AREA_TOP`
  - `CUSTOM_AREA_RIGHT`
  - `CUSTOM_AREA_BOTTOM`
- Functions
  - `showCustomArea(area)`—Specifies the custom area to display. By default custom areas are hidden.

    Parameter:
    - `area`—Specifies the area to display. The value is one of the Constants.
  - `hideCustomArea(area)`—Specifies the custom area to hide.

    Parameter:
    - `area`—Specifies the area to hide. The value is one of the Constants.
  - `setCustomAreaSize(area, size)`

    Parameters:
    - `area`—Specifies the area to display. The value is one of the Constants.
    - `size`—Specifies the height or the width of the area in pixels. If the `setCustomAreaSize` parameter is not called, the default size is 19 pixels.
  - `getCustomAreaFrameName(area)`—Returns the name of the frame that is used to display the specified custom area.

    Parameter:
    - `area`—Specifies the area to display. The value is one of the Constants.

1. Create the JavaScript file `htmlcontrol.js` (refer to “htmlcontrol.js” on page 100).
2. (Optional) Create the Custom Area Top DHTML page `custom/custom-area-top.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html; charset=utf-8" %>
<%
    String userNameParam = request.getParameter("userName");
```

```
%>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="style.css"/>
    <script language="javascript" src="htmlcontrol.js"></script>
   <script language="javascript">
      window.parent.includeGDFunctionalities(window);

      function custom_load() {
        activateGDFunctionalities();
        addControl(custom_closeButton);
        focusOnControl(custom_closeButton);
      }

      function custom_unload() {
        unactivateGDFunctionalities();
      }

      function custom_onCloseButton() {
        window.parent.hideCustomArea(window.parent.CUSTOM_AREA_TOP);
      }
    </script>
  </head>
  <body width="100%" height="100%" onLoad="custom_load()" onUnload="custom_unload()">
    <table class="bgColorTop" width="100%" height="100%" border="0" cellpadding="0"
        cellspacing="0">
      <tr width="100%">
        <td class="whiteBorder" width="1px" rowspan="5"><img alt="" src="empty.gif"
          border="0" width="1px"></td>
        <td class="whiteBorder" height="1px" colspan="4"><img alt="" src="empty.gif"
          border="0" height="1px"></td>
      </tr>
      <tr>
        <td width="2px" rowspan="3"><img alt="" src="empty.gif" border="0"
          width="2px"></td>
        <td height="2px"><img alt="" src="empty.gif" border="0" height="2px"></td>
        <td width="2px" rowspan="3"><img alt="" src="empty.gif" border="0"
          width="2px"></td>
        <td class="grayBorder" width="1px" rowspan="4"><img alt="" src="empty.gif"
          border="0" width="1px"></td>
      </tr>
      <tr width="100%" height="100%">
        <td width="100%" height="100%">
          <table width="100%" height="100%" border="0" cellpadding="0" cellspacing="0">
            <tr width="100%" height="100%">
              <td class="fontName labelText boldText" width="100%" align="center"
                valign="middle">TOP</td>
              <td><div id="closeDiv"></div></td>
            </tr>
          </table>
        </td>
```

```
      </tr>
      <tr>
        <td height="2px"><img alt="" src="empty.gif" border="0" height="2px"></td>
      </tr>
      <tr>
        <td class="grayBorder" height="1px" colspan="3"><img alt="" src="empty.gif"
   border="0" height="1px"></td>
      </tr>
    </table>
    <script language="javascript">
      var custom_closeButton = custom_createHTMLButtonImage ("closeDiv", "Close",
        ["CloseBtn.gif","CloseBtng.gif","CloseBtnd.gif"], 25,
        "custom_onCloseButton()");
    </script>
  </body>
</html>
```

3. (Optional) Create the Custom Area Right DHTML page `custom/custom-area-right.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html; charset=utf-8" %>
<%
    String userNameParam = request.getParameter("userName");
%>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="style.css"/>
    <script language="javascript" src="htmlcontrol.js"></script>
    <script language="javascript">
      window.parent.includeGDFunctionalities(window);

      function custom_load() {
        activateGDFunctionalities();
        addControl(custom_leftButton);
        addControl(custom_rightButton);
      }

      function custom_unload() {
        unactivateGDFunctionalities();
      }

      var custom_currentSize = 100;

      function custom_onArrowButton(newSize) {
        if (newSize == custom_currentSize)
          return;

        window.parent.setCustomAreaSize(window.parent.CUSTOM_AREA_RIGHT, newSize);
        custom_currentSize = newSize;
        if (custom_currentSize == 100) {
```

```
          custom_leftButton.enable();
          custom_rightButton.disable();
        } else {
          custom_leftButton.disable();
          custom_rightButton.enable();
        }
      }
    </script>
  </head>
  <body width="100%" height="100%" onLoad="custom_load()" onUnload="custom_unload()">
    <table class="bgColorRight" width="100%" height="100%" border="0" cellpadding="0"
        cellspacing="0">
      <tr width="100%">
        <td class="whiteBorder" width="1px" rowspan="5"><img alt="" src="empty.gif"
           border="0" width="1px"></td>
        <td class="whiteBorder" height="1px" colspan="4"><img alt="" src="empty.gif"
           border="0" height="1px"></td>
      </tr>
      <tr>
        <td width="2px" rowspan="3"><img alt="" src="empty.gif" border="0"
           width="2px"></td>
        <td height="2px"><img alt="" src="empty.gif" border="0" height="2px"></td>
        <td width="2px" rowspan="3"><img alt="" src="empty.gif" border="0"
           width="2px"></td>
        <td class="grayBorder" width="1px" rowspan="4"><img alt="" src="empty.gif"
           border="0" width="1px"></td>
      </tr>
      <tr width="100%" height="100%">
        <td width="100%" height="100%">
          <table border="0" cellpadding="0" cellspacing="0" align="center"
             valign="middle">
            <tr width="100%" height="100%">
              <td><div id="leftDiv"></div></td>
              <td rowspan="2"><img alt="" src="empty.gif" border="0" width="5px"></td>
              <td class="fontName labelText boldText" width="100%" align="center"
                valign="middle" rowspan="2">RIGHT</td>
              <td rowspan="2"><img alt="" src="empty.gif" border="0" width="5px"></td>
            </tr>
            <tr width="100%" height="100%">
              <td><div id="rightDiv"></div></td>
            </tr>
          </table>
        </td>
      </tr>
      <tr>
        <td height="2px"><img alt="" src="empty.gif" border="0" height="2px"></td>
      </tr>
      <tr>
        <td class="grayBorder" height="1px" colspan="3"><img alt="" src="empty.gif"
           border="0" height="1px"></td>
      </tr>
```

```
    </table>
    <script language="javascript">
      var custom_leftButton = custom_createHTMLButtonImage ("leftDiv", "Left",
        ["LeftArrowBtn.gif","LeftArrowBtnDisable.gif","LeftArrowBtnFocused.gif"], 25,
        "custom_onArrowButton(300)");
      var custom_rightButton = custom_createHTMLButtonImage ("rightDiv", "Right",
        ["RightArrowBtn.gif","RightArrowBtnDisable.gif","RightArrowBtnFocused.gif"],
        25, "custom_onArrowButton(100)");
      custom_rightButton.disable();
    </script>
  </body>
</html>
```

4. (Optional) Create the Custom Area BottomDHTML page `custom/custom-area-bottom.jsp`:

```
<%@ page language="java" buffer="none" contentType="text/html; charset=utf-8" %>
<%
    String userNameParam = request.getParameter("userName");
%>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="style.css"/>
    <script language="javascript">
      window.parent.includeGDFunctionalities(window);

      function custom_load() {
        activateGDFunctionalities();
      }

      function custom_unload() {
        unactivateGDFunctionalities();
      }
    </script>
  </head>
  <body width="100%" height="100%" onLoad="custom_load()" onUnload="custom_unload()">
    <table class="bgColorBottom" width="100%" height="100%" border="0" cellpadding="0"
     cellspacing="0">
      <tr width="100%">
        <td class="whiteBorder" rowspan="5" width="1px"></td>
        <td class="whiteBorder" height="1px" colspan="4"></td>
      </tr>
      <tr height="2px">
        <td width="2px" rowspan="3"></td>
        <td></td>
        <td width="2px" rowspan="3"></td>
        <td class="grayBorder" width="1px" rowspan="4"></td>
      </tr>
      <tr width="100%" height="100%">
```

```
        <td class="fontName labelText boldText" align="center"
   valign="middle">BOTTOM</td>
      </tr>
      <tr height="2px">
        <td></td>
      </tr>
      <tr height="1px">
        <td class="grayBorder" colspan="3"></td>
      </tr>
    </table>
  </body>
</html>
```

5. Create the customization file `custom/custom.xml`:

```
<gcn-resources>
    <desktop>
        <javascript-onload>
            <![CDATA[
                document.forms.customAreaTopForm.target =
   getCustomAreaFrameName(CUSTOM_AREA_TOP);
                document.forms.customAreaTopForm.elements.userName.value = userName;
                document.forms.customAreaTopForm.submit();

                document.forms.customAreaRightForm.target =
   getCustomAreaFrameName(CUSTOM_AREA_RIGHT);
                document.forms.customAreaRightForm.elements.userName.value = userName;
                document.forms.customAreaRightForm.submit();

                document.forms.customAreaBottomForm.target =
   getCustomAreaFrameName(CUSTOM_AREA_BOTTOM);
                document.forms.customAreaBottomForm.elements.userName.value = userName;
                document.forms.customAreaBottomForm.submit();

                showCustomArea(CUSTOM_AREA_TOP);
                showCustomArea(CUSTOM_AREA_RIGHT);
                setCustomAreaSize(CUSTOM_AREA_RIGHT, 100);

                showCustomArea(CUSTOM_AREA_BOTTOM);
                checkMenuBarItemByName("bottom");
            ]]>
        </javascript-onload>

        <html-body>
            <![CDATA[
                <form name="customAreaTopForm" target="_blank" method="get"
                   action="custom/custom-area-top.jsp">
                    <input type="hidden" name="userName"/>
                </form>
```

```
                <form name="customAreaRightForm" target="_blank" method="get"
                   action="custom/custom-area-right.jsp">
                    <input type="hidden" name="userName"/>
                </form>
                <form name="customAreaBottomForm" target="_blank" method="get"
                   action="custom/custom-area-bottom.jsp">
                    <input type="hidden" name="userName"/>
                </form>
                <script>
                    var bottomAreaShown = true;
                </script>
            ]]>
        </html-body>

        <menu-bar>
            <display>true</display>
            <menus>
                <menu name="custom">
                    <text>Custom</text>
                    <menu-items>
                        <menu-item name="bottom">
                            <text>Bottom Area</text>
                            <checkable>true</checkable>
                            <javascript-onselect>
                                <![CDATA[
                                    if (bottomAreaShown) {
                                        bottomAreaShown = false;
                                        hideCustomArea(CUSTOM_AREA_BOTTOM);
                                        uncheckMenuBarItemByName("bottom");
                                    } else {
                                        bottomAreaShown = true;
                                        showCustomArea(CUSTOM_AREA_BOTTOM);
                                        checkMenuBarItemByName("bottom");
                                    }
                                ]]>
                            </javascript-onselect>
                        </menu-item>
                    </menu-items>
                </menu>
            </menus>
        </menu-bar>
    </desktop>
</gcn-resources>
```

Customization Template

This section presents an example of an XML customization file, which you can use as a template.

The section contains the following sub-sections:

Customization Guidelines

To use this template to create an actual customization file, replace all occurrences of the following place-holders:

- `boolean`—Replace with `true` or `false`.
- `code`—Replace with some JavaScript code.
- `html`—Replace with some DHTML code.
- `path`—Replace with the path to a filename.
- `string`—Replace with a string of characters.

The content that replaces `boolean`, `path`, and `string` must not contain any space characters in their enclosing tag.

The content that replaces `html` and `code` should be enclosed in a `<![CDATA[]]>` tag.

Also, you can repeat the following sections as many times as needed in their containing sections:

- `<button>`
- `<menu>`
- `<menu-item>`
- `<shortcut>`
- `<tab>`
- `<tool-bar>`

Customization Template File

```
<gcn-resources>
   <media>
      <chat>boolean</chat>
      <e-mail>boolean</e-mail>
      <voice>boolean</voice>
      <co-browse>boolean</co-browse>
      <im>boolean</im>
      <webcallback>boolean</webcallback>
      <smssession>boolean</smssession>
      <open-medias>
         <open-media>
```

```
        <activate>boolean</ activate>
        <open-media-type>Media Type</open-media-type>
      </open-media>
    </open-medias>
  </media>
  <desktop>
    <dictionary-class>path</dictionary-class>
    <javascript-onload>
      code
    </javascript-onload>
    <javascript-onunload>
      code
    </javascript-onunload>
    <javascript-onaddinteraction>
      code
    </javascript-onaddinteraction>
    <javascript-onremoveinteraction>
      code
    </javascript-onremoveinteraction>
    <javascript-onselectinteraction>
      code
    </javascript-onselectinteraction>
    <javascript-oncustomevent>
      code
    </javascript-oncustomevent>
    <html-header>
      html
    </html-header>
    <html-body>
      html
    </html-body>
    <status-bar>
      <display>boolean</display>
    </status-bar>
    <menu-bar>
      <display>boolean</display>
      <menus>
        <javascript-onload>
          code
        </javascript-onload>
        <javascript-onunload>
          code
        </javascript-onunload>
        <html-header>
          html
        </html-header>
        <html-body>
          html
        </html-body>
        <menu name="string"> <!-- (repeatable) -->
          <text>string</text>
```

```
            <dictionary-key>string</dictionary-key>
            <menu-items>
               <menu-item name="string">
                  <!-- (menu-item is repeatable) -->
                  <text>string</text>
                  <dictionary-key>string</dictionary-key>
                  <checkable>boolean</checkable>
                  <javascript-onload>
                     code
                  </javascript-onload>
                  <javascript-onunload>
                     code
                  </javascript-onunload>
                  <javascript-onselect>
                     code
                  </javascript-onselect>
                  <html-header>
                     html
                  </html-header>
                  <html-body>
                     html
                  </html-body>
               </menu-item>
            </menu-items>
         </menu>
      </menus>
   </menu-bar>
   <tool-bar>
      <display>boolean</display>
      <tool-bar-items>
         <javascript-onload>
            code
         </javascript-onload>
         <javascript-onunload>
            code
         </javascript-onunload>
         <html-body>
            html
         </html-body>
         <html-header>
            html
         </html-header>
         <tool-bar-item name="string">
            <!-- (tool-bar-item is repeatable) -->
            <tooltip-text>
               string
            </tooltip-text>
            <tooltip-dictionary-key>
               string
            </tooltip-dictionary-key>
            <image-up-name>
```

```
            path
          </image-up-name>
          <image-down-name>
            path
          </image-down-name>
          <image-disable-name>
            path
          </image-disable-name>
          <image-focus-name>
            path
          </image-focus-name>
          <javascript-onload>
            code
          </javascript-onload>
          <javascript-onselect>
            code
          </javascript-onselect>
          <javascript-onunload>
            code
          </javascript-onunload>
          <html-body>
            html
          </html-body>
          <html-header>
            html
          </html-header>
        </tool-bar-item>
      </tool-bar-items>
    </tool-bar>
    <menu-organizer>
      <display>boolean</display>
    </menu-organizer>
    <shortcuts>
      <shortcut name="string">
        <text>
          string
        </text>
        <dictionary-key>
          string
        </dictionary-key>
        <javascript-onexecute>
          code
        </javascript-onexecute>
      </shortcut>
    </shortcuts>
    <interaction-action>
      <javascript-onload>
        code
      </javascript-onload>
      <javascript-onunload>
        code
```

```
    </javascript-onunload>
    <javascript-oncustomevent>
      code
    </javascript-oncustomevent>
    <html-header>
      html
    </html-header>
    <html-body>
      html
    </html-body>
    <buttons>
      <button name="string"> <!-- (repeatable) -->
        <tooltip-text>
          string
        </tooltip-text>
        <tooltip-dictionary-key>
          string
        </tooltip-dictionary-key>
        <image-up-name>path</image-up-name>
        <image-disable-name>path</image-disable-name>
        <image-focus-name>path</image-focus-name>
        <javascript-onload>
          code
        </javascript-onload>
        <javascript-onunload>
          code
        </javascript-onunload>
        <javascript-onselect>
          code
        </javascript-onselect>
        <javascript-oninteractionstatuschange>
          code
        </javascript-oninteractionstatuschange>
        <html-header>
          html
        </html-header>
        <html-body>
          html
        </html-body>
      </button>
    </buttons>
  </interaction-action>
  <interaction-information>
    <display>boolean</display>
    <javascript-onload>
      code
    </javascript-onload>
    <javascript-onunload>
      code
    </javascript-onunload>
    <javascript-oncustomevent>
```

```
      code
    </javascript-oncustomevent>
    <html-header>
      html
    </html-header>
    <javascript-onappendinformationstatechange>
      code
    </javascript-onappendinformationstatechange>
    <html-body>
      html
    </html-body>
    <tabs>
      <tab name="string"> <!-- (repeatable) -->
        <text>string</text>
        <dictionary-key>string</dictionary-key>
        <scrollbar>boolean</scrollbar>
        <javascript-onload>
          code
        </javascript-onload>
        <javascript-onunload>
          code
        </javascript-onunload>
        <javascript-onselect>
          code
        </javascript-onselect>
        <javascript-oninteractionstatuschange>
          code
        </javascript-oninteractionstatuschange>
        <html-header>
          html
        </html-header>
        <html-body>
          html
        </html-body>
      </tab>
    </tabs>
  </interaction-information>
  <resources>
    <display>boolean</display>
    <javascript-onload>
      code
    </javascript-onload>
    <javascript-onunload>
      code
    </javascript-onunload>
    <javascript-oncustomevent>
      code
    </javascript-oncustomevent>
    <javascript-onappendinformationstatechange>
      code
    </javascript-onappendinformationstatechange>
```

```
    <html-header>
      html
    </html-header>
    <html-body>
      html
    </html-body>
    <tabs>
      <tab name="string"> <!-- (repeatable) -->
      <text>string</text>
      <dictionary-key>string</dictionary-key>
      <scrollbar>boolean</scrollbar>
      <javascript-onload>
        code
        </javascript-onload>
        <javascript-onunload>
          code
        </javascript-onunload>
        <javascript-onselect>
          code
        </javascript-onselect>
        <html-header>
          html
        </html-header>
        <html-body>
          html
        </html-body>
      </tab>
    </tabs>
  </resources>
  <customer-records>
    <display>boolean</display>
    <javascript-onload>
      code
    </javascript-onload>
    <javascript-onunload>
      code
    </javascript-onunload>
    <javascript-oncustomevent>
      code
    </javascript-oncustomevent>
    <javascript-onappendinformationstatechange>
      code
    </javascript-onappendinformationstatechange>
    <html-header>
      html
    </html-header>
    <html-body>
      html
    </html-body>
    <tabs>
      <tab name="string"> <!-- (repeatable) -->
```

```
                    <text>string</text>
                    <dictionary-key>string</dictionary-key>
                    <scrollbar>boolean</scrollbar>
                    <javascript-onload>
                      code
                    </javascript-onload>
                    <javascript-onunload>
                      code
                    </javascript-onunload>
                    <javascript-onselect>
                      code
                    </javascript-onselect>
                    <html-header>
                      html
                    </html-header>
                    <html-body>
                      html
                    </html-body>
                  </tab>
                </tabs>
              </customer-records>
            </desktop>
          </gcn-resources>
```

Chapter

3 Open Media Extensions

This chapter outlines the extension procedures for the Genesys Desktop 7.6 GUI Layer.

This chapter contains the following sections:

Overview

To customize the display for Open Media interactions in Genesys Agent Desktop GUI, you create an `interaction.xml` extension file that Genesys Desktop loads at startup; in some cases, you can additionally edit a JSP file or certain other files, such as HTML files.

Extension code is kept separate from Genesys Desktop display code, in order to simplify Genesys Desktop update releases and corresponding migration procedures.

Extension code is a set of simple tags that describe GUI areas of the Genesys Agent Desktop. It abstracts most of the action code and enables to specify the following items:

- Images
- Buttons
- Menus
- Toolbars
- Views

The extension XML file describes the elements to be added and abstracts some DHTML code. The Genesys Desktop Server loads this file when it starts up, and dynamically adds your extension code to the regular Genesys Agent Desktop code, as shown in Figure 14.

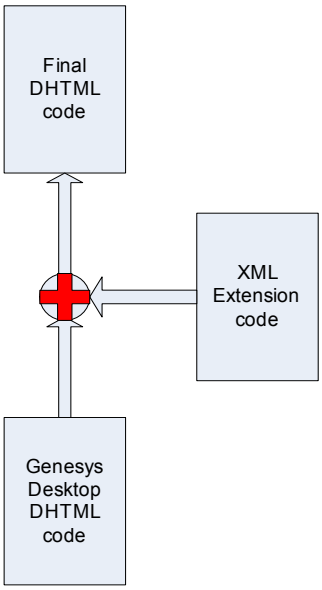


Figure 14: Dynamic Merging of Extension code

Deploying Your Extended Genesys Agent Desktop

To extend your Genesys Agent Desktop, first create a directory named `extension` in the following directory:
`[install]/webapps/gdesktop/`

For each type of Open Media interaction that your extended Genesys Agent Desktop application should support, create an `<Open Media Extension>` directory in the `extension` directory.

Then, in the `<Open Media Extension>` directory, create an `interaction.xml` file that conforms to the syntax described in “Principles to Write an Open Media Extension” on page 81.

Also, in the `<Open Media Extension>` directory, copy all the resources that your extension needs:

- Image files
- JSP, HTML, DHTML pages

This is all you need to deploy extensions. At application startup, Genesys Desktop parses the content of the `gdesktop/extension/` subdirectories and loads all the extensions found.

Principles to Write an Open Media Extension

To write an Open Media extension file, use the tags listed in the "Extension Tags for Open Media Interactions" section of the *Genesys Desktop 7.6 XML Tag Reference.*

This section presents the main tags you will use to write an Open Media extension.

Defining an Interaction Filter

When Genesys Desktop loads the `interaction.xml` file, it uses the filter to determine for which type of Open Media interaction the Agent GUI is extended. The filter takes into account interaction attribute values defined in the Configuration Layer:

- `Media Type`
- `Interaction Type`
- `Interaction Subtype`

The `<filter>` tag encloses interaction attribute values as key-value pairs. The following filter extends the Genesys Desktop for `sms` interactions of type `Inbound` and subtype `InboundNew`:

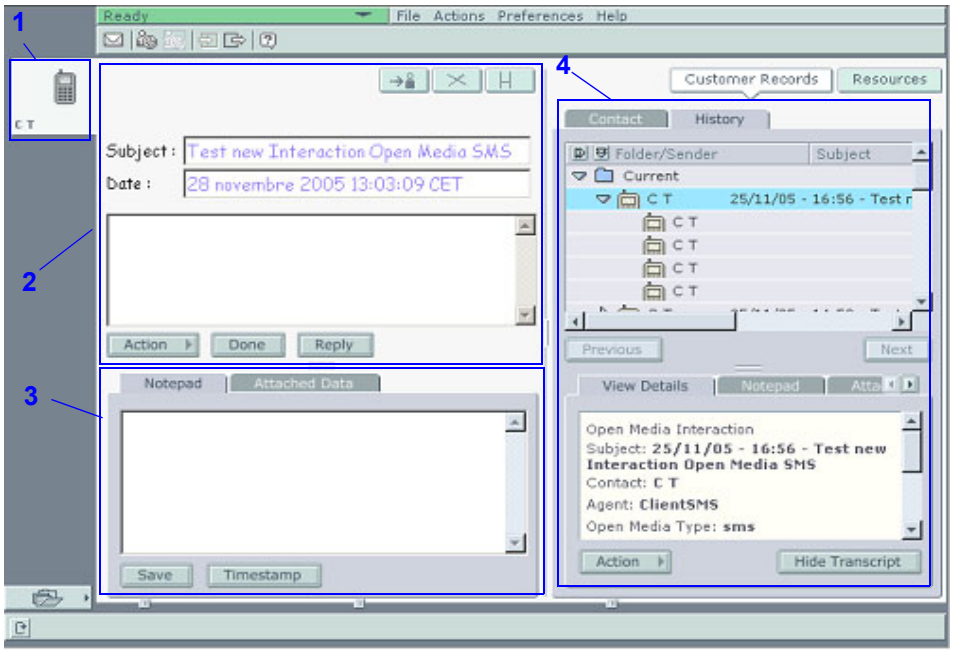
```
<filter>
   <parameters>
      <parameter>
         <key>Interaction.OpenMediaType</key>
         <value>sms</value>
      </parameter>
      <parameter>
         <key>Interaction.OpenInteractionType</key>
         <value>Inbound</value>
      </parameter>
      <parameter>
         <key>Interaction.OpenInteractionSubtype</key>
         <value>InboundNew</value>
      </parameter>
```

```
   </parameters>
</filter>
```

Note: The generic extension has the value `any` for the `Interaction.OpenMediaType`. To use it, replace `any` with a value that represents a specific media type.

Set up GUI Areas

You extend Genesys Desktop by specifying the content and behavior of specific GUI areas. Figure 15 presents the Agent Desktop GUI areas that an extension can define.



1. Navigation bar
2. Interaction area
3. Interaction information
4. Workbin and history area

Figure 15: Extendable GUI Areas

Each area is associated with a tag that encloses the specification of the GUI components to be added. Table 1 presents the main tags that define area contents.

Table 1: Tags and Areas

Area	Tag	Tag Contents
Navigation bar	`<navigation-bar>`	Specifies an icon for the Open Media interaction in the navigation bar.
Interaction area	`<interaction-area>`	Displays the selected Open Media interaction.
Interaction information	`<interaction-information>`	Describes additional information about the selected Open Media interaction
Workbin and history	`<workbin>` `<history>`	Specifies interaction images and information to be displayed in workbins and histories.

Set up Images for Navigation and Lists

To set up an image in the navigation bar or in the interaction lists, first copy the icon file into a `gdesktop/extension/` subdirectory.

The following code example shows the syntax used to specify an icon for the `sms` Open Media interaction in the navigation bar.

```
<navigation-bar>
<interaction-image>extension/sms_directory/sms-batch.gif
</interaction-image>
</navigation-bar>
```

In workbin and history components, you can set images for the interactions displayed. In particular, the `<history>` tag allows you to distinguish pending interactions from non-pending interactions by setting a different image according to the interaction status, as shown in the following code snippet.

```
<history>
   <interaction-image>extension/sms_directory/sms-img.gif</interaction-image>
   <interaction-pending-image>extension/sms_directory/sms-img-pending.gif
   </interaction-pending-image>
<interaction-detail>
   ...
</interaction-detail>
</history>
```

Set up Pages

You can load HTML and JSP pages in tabs and in the interaction area. To do so, use the `<view>` tag, as shown in the following code snippet.

```
<view>
        <url>extension/sms_directory/sms-in.jsp</url>
</view>
```

Figure 16 shows the `sms-in.jsp` page loaded in the media view.

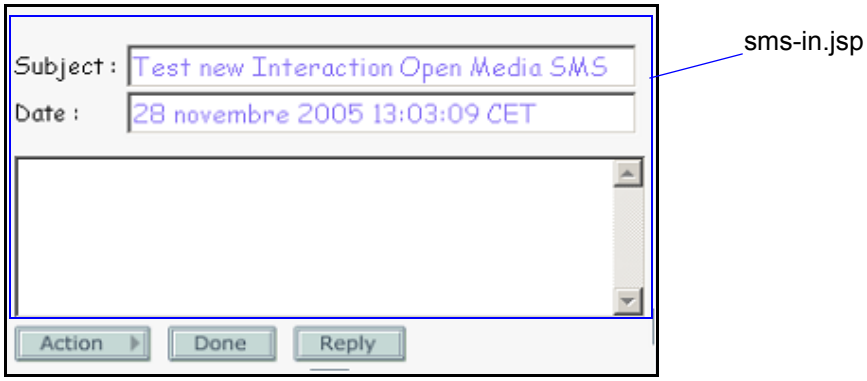


Figure 16: Extended View

Set up Tabs

In the `<interaction-information>` and `<interaction-detail>` tags, you can set up contents of new tabs, and also contents of predefined tabs, such as `Attached Data`, `View Details`, or `Notepad`.

The `<tab>` tag encloses the tag content to which the following components can be added:

- Buttons enclosed by the `<buttons>` tag.
- Menu items enclosed by the `<menu-items>` tag to the action menu.
- A content page to display.

The following code snippet shows an extension code that defines a `Detail` tab and adds two predefined tabs to the interactions list detail area.

```
<interaction-detail>
   <tabs>
      <tab>
         <name>Detail</name>
         <text>Detail</text>
         <action>
            <url>extension/sms-in/sms-in-detail.jsp</url>
         </action>
         <buttons>
            <button>
               <text>Action</text>
               <menu-items>
                  <menu-item>
                     <name>ACTION_NEW_PHONE_CALL</name>
                  </menu-item>
                  <menu-item>
```

```
                    <name>ACTION_NEW_EMAIL</name>
                  </menu-item>
                </menu-items>
                <name>action</name>
              </button>
            </buttons>
          </tab>
          <tab>
            <name>TAB_NOTEPAD</name>
          </tab>
          <tab>
            <name>TAB_ATTACHED_DATA</name>
          </tab>
        </tabs>
</interaction-detail>
```

For further details about predefined tabs, see the *Genesys Desktop 7.6 XML Tag Reference*.

Set up Actions

You can set up actions for buttons, button images, menus and tabs. Actions abstract HTML code, and can be either predefined actions or calls to pages (JSP, DHTML, or HTML).

To define a new action, you indicate which page should be executed by using the `<url>` tag, and how to handle the result by using the `<target>` tag.

The following code example defines a help button that loads the `help.jsp` file and displays the result in a pop-up window.

```
<button>
  <name>help</name>
  <action>
    <url>extension/sms/help.jsp</url>
    <target>
      <type>TARGET_POPUP</type>
      <parameters>
      <parameter><key>menubar</key><value>no</value></parameter>
      <parameter><key>status</key><value>no</value></parameter>
      <parameter><key>titlebar</key><value>no</value></parameter>
      <parameter><key>toolbar</key><value>no</value></parameter>
      </parameters>
    </target>
  </action>
</button>
```

To use predefined actions, you specify a constant with the `<name>` tag, as shown here:

```
<action>
   <name>ACTION_MARK_DONE</name>
</action>
```

Open Media Workflow

Your extension should enable the agent to display and/or process Open Media interactions. Depending on the GUI components added to Genesys Desktop, the agent should be able to process this interaction. For instance, the agent should be able to:

- add to or modify the information for the interaction.
- reply to and send the Open Media interaction.
- check the spelling of the Open Media interaction

Automatic Save

When you set up new actions, you enable the automatic save by using the `<save>` child tag. If the automatic save is enabled, or if you use a predefined action including an automatic save, Genesys Desktop saves the information specified by the `<save-action>` tag before performing the action.
In the following code snippet, the information to be saved corresponds to the Open Media interaction text and subject, stored in some JavaScript variables available in the extension JavaScript context:

```
<interaction-area>
   ...
      ...
         <action>
            <name>MyAction</name>
            <save>true</save>
         </action>
   ...
   <save-action>
      <parameters>
         <parameter><key>Interaction.Subject</key></parameter>
         <parameter><key>Interaction.Text</key></parameter>
      </parameters>
   </save-action>
   ...
</interaction-area>
```

Send the Open Media Interaction

To send an Open Media interaction, your extension must:

- Add an `ACTION_SEND` predefined action to a GUI component, for instance, a button or a menu item.
- Specify which information must be saved before sending the Open Media interaction.

- Use the <send-action> tag and set the queue name to which the interaction is submitted.

The following example defines a send action for an sms in Open Media interaction.

```
<interaction-area>
   <interaction-primary-controls>
      <buttons>
         <button>
            <name>ACTION_SEND</name>
         </button>
      </buttons>
   <interaction-primary-controls>
   <save-action>
      ...
   </save-action>
   <send-action>
      <queue>
         {Application.Options.multimedia.sms-outbound-queue}
      </queue>
   </send-action>
</interaction-area>
```

Reply to an Open Media Interaction

As with e-mail interactions, when replying to Open Media interactions, Genesys Desktop creates a new Open Media interaction for the reply. You must specify which type of Open Media interaction should be used.

The extension must define at least the following interaction properties:

- Media Type
- Interaction Type
- Interaction Subtype

These properties must be consistent with interaction attribute values set in the Configuration Layer.

To define this reply, your extension must:

- Add an ACTION_REPLY predefined action to a GUI component, for instance, a button or a menu item.
- Use the <reply-action> tag to define the creation parameters of the replying interaction.

The following example defines a reply action for an sms in Open Media interaction.

```
<interaction-area>
   <interaction-primary-controls>
      <buttons>
         <button>
```

```
            <name>ACTION_REPLY</name>
         </button>
      </buttons>
   </interaction-primary-controls>
   <reply-action>
      <parameters>
         <parameter>
            <key>NewInteraction.OpenMediaType</key>
            <value>sms</value>
         </parameter>
         <parameter>
            <key>NewInteraction.OpenInteractionType</key>
            <value>Outbound</value>
         </parameter>
         <parameter>
            <key>NewInteraction.OpenInteractionSubtype</key>
            <value>OutboundReply</value>
         </parameter>
         <parameter>
            <key>NewInteraction.Subject</key>
            <value>Re: {Interaction.Subject}</value>
         </parameter>
      </parameters>
      <queue>
         {Application.Options.multimedia.open-media-outbound-queue}
      </queue>
   </reply-action>
</interaction-area>
```

Note: You can use object properties `{property_name}` for parameter values. See *Genesys Desktop 7.6 XML Tag Reference* for further details.

Disposition Code

When you set up new actions, you can define dispostion code selection as mandatory prior to applying the new action by using the following child tag: `<disposition-code-mandatory>`

The Disposition Code feature is available for actions that are done in the following tags: `interaction-secondary-controls` and `interaction-primary-controls`.

By default, this mechanism is not applied, except for the following predefined actions (which are associated with this mechanism):

- ACTION_MARK_DONE
- ACTION_STOP
- ACTION_SEND

The following example defines a custom action that is associated to the mandatory mechanism:

```
<interaction-area>
    <interaction-primary-controls>
        <buttons>
            <button>
                <text>My Custom Action</text>
                <action>
                    <url>my-action.jsp</url>
                    <disposition-code-mandatory>true</disposition-code-mandatory>
                </action>
            </button>
        </buttons>
    </interaction-primary-controls>
</interaction-area>
```

Note: The Disposition Code mechanism is not taken into account for targets of type `TARGET_POPUP`.

Spelling Check

To check the spelling of an Open Media interaction, your extension must be able to do the following functions:

- Add an ACTION_SPELL_CHECKER predefined action to a GUI component, for example, a button or a menu item.
- Use the `<spell-check-action>` tag to define the fields to have the spelling checked.
- Use the `<auto-spell-check>` tag to specify whether to apply or not the auto spelling check function for a particular action.

The following example defines a Spell Check action for a sms in an Open Media interaction.

```
        <interaction-primary-controls>
            <buttons>
...
                <button>
                    <name>ACTION_SEND</name>
                    <auto-spell-check>true</auto-spell-check>
                </button>
...
        </interaction-primary-controls>
...
        <spell-check-action>
          <spell-check-fields>
            <spell-check-field>
              <spell-check-field-body>Interaction.Subject</spell-check-field-body>
              <spell-check-field-label>Subject.Label</spell-check-field-label>
            </spell-check-field>
            <spell-check-field>
```

```
            <spell-check-field-body>Interaction.Text</spell-check-field-body>
            <spell-check-field-format>Text.Format</spell-check-field-format>
          </spell-check-field>
        </spell-check-fields>
      </spell-check-action>
    </interaction-area>
```

JavaScript Extension Context

The JavaScript extension context stores variables and the corresponding values in a data array. You can get and set these variables, and use this array in your own JSP and DHTML pages.

The following JavaScript uses of this array of the Agent Interaction Layer library to store the text and subject of an Open Media interaction.

```
<script language="JavaScript" >
   var extensionContext = window.parent.getExtensionContext();

function load() {
   var elements = document.forms.smsOutForm.elements;
   var value = extensionContext["Interaction.Subject"];
   if (value != null) {
      elements.subjectField.value = value;
   } else {
      extensionContext["Interaction.Subject"] = "<%= strSubject %>";
   }
   value = extensionContext["Interaction.Text"];
   if (value != null) {
      elements.textField.value = value;
   } else {
      extensionContext["Interaction.Text"] = "<%= strMessage %>";
   }
   updateCounters();
}
```

In your customization, you can access the AIL classes in JSPs or servlets. If you instantiate the `AilFactory` object, you access the same factory used in Genesys Desktop, which is a singleton (see Figure 5 on page 29).

JSP and DHTML files called in your extension code can access the Agent Interaction Layer to retrieve more information regarding the Open Media interaction, or to take some action.

For further details about Agent Interaction Layer, see “Genesys Desktop Calls the Agent Interaction Layer” on page 29.

Insert SRL & External Knowledge Base in an Open Media Interaction

Genesys Desktop enables agents to insert Standard Response Library (SRL) and external knowledge base information into an interaction. You can add this capability to your Open-Media customization.

Use the `javascript-onappendinformation` child tag of the `interaction-area` tag to insert your custom JavaScript code. By default the action is disabled.

The `javascript-onappendinformation` tag uses the `information` variable to append information. It is a plain text variable by default; however, if the `acceptHTMLFormat` JavaScript function has been called, then HTML format is supported.

Some JavaScript functions are provided in this area usable by custom pages:

- `acceptHTMLFormat`—Authorize the HTML format. Call this function in the `onload callback` of the custom page.
- `enableAppendInformation`—Enables insert buttons for SRL or an external knowledge base. Call this function in the `onload callback` of the custom page.
- `disableAppendInformation`—Disables insert buttons for SRL or an external knowledge base. Call this function in the `onUnLoad callback` of the custom page.
- `isPossibleAppendInformation`—Informs whether the insert is possible in the current interaction.
- `getDetailFrameName`—Returns the name of the frame in which the detail of the interaction is displayed.

Contact Selection in an Open Media Interaction

Genesys Desktop enables agents to select a contact for some types of interactions, such as outbound e-mail.

The following JavaScript functions are provided in this area that are usable by custom pages:

- `selectContact`—Replaces the interaction area by a panel that enables contact selection. After the contact is selected, the `idContact` is provided on HTTP request for the view display (specified in the view tag). The contact is not associated to the interaction, it is in charge of the custom code. This should be done in the JSP that is used for the view.
- `isUCSInService`—Informs if the Universal Contact Server (UCS) is in service.

The following JavaScript callback can be implemented by custom pages:

- `onUCSBackInService`—Callback informs when Universal Contact Server (UCS) is back in service.
- `onUCSOutOfService`—Callback informs when the Universal Contact Server(UCS) is out of service.

Agent Desktop Customization Propagated in Open Media Interactions

Tags and sub-tags from `<interaction-action>` and `<interaction-information>` that come from the Agent Desktop Customization can interact with Open-Media customization.

It is possibe to do the following functions:

- Add buttons in the interaction-secondary-controls area by using the `<interaction-action>` tag
- Add tabs in the interaction-information area by using the `<interaction-information>` tag
- Add HTML and JavaScript code in the interaction-secondary-controls area by using the `<interaction-action>` tag
- Add HTML and JavaScript code in the interaction-information area by using the `<interaction-information>` tag
- Use JavaScript functions that are available for the Agent Desktop Customization on objects that are related to the Open-Media customization

When buttons or tabs are configured in the Agent Desktop Customization and the Open-Media customization, they are mixed.

For buttons in the interaction area, buttons from Agent Desktop Customization are displayed to the left of buttons from the Open-Media customization. For tabs in the interaction information area, tabs from Agent Desktop Customization are displayed to the right of tabs from the Open-Media customization. Refer to Figure 17 on page 93.

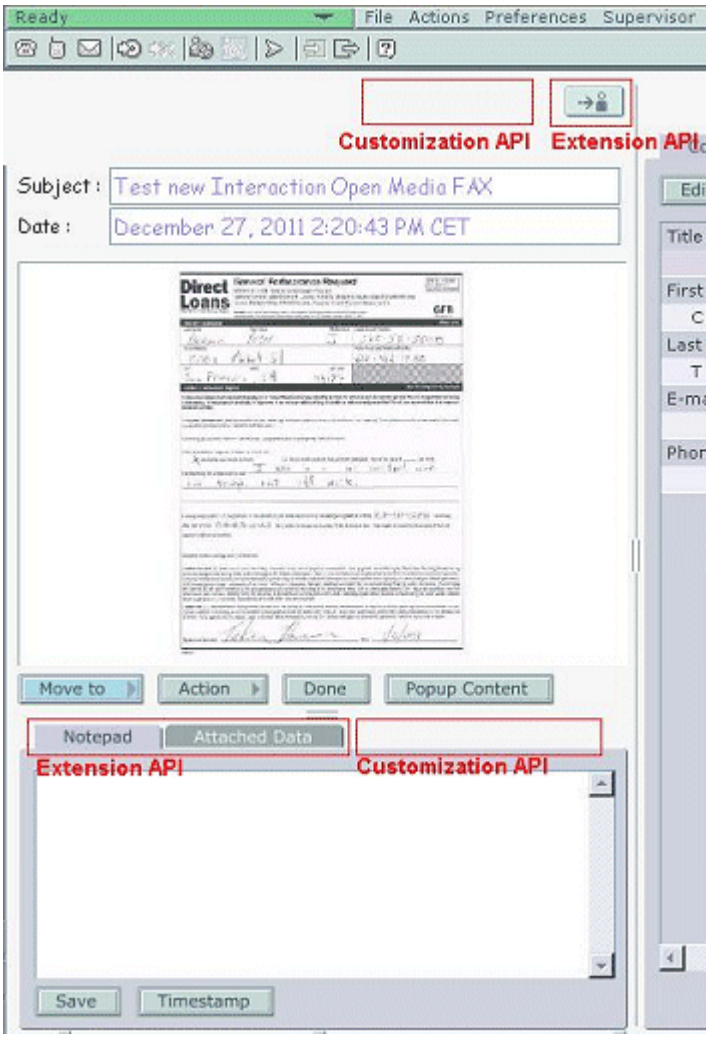


Figure 17: Agent Desktop customization propagated in an Open Media customization

Limitations and Restrictions

Please note the following limitations and restrictions as you customize Genesys Agent Desktop:

- Only one AIL runs in the Genesys Desktop server; therefore, your custom code shares the same instance as the Genesys Agent Desktop GUI Layer, and has access to the same objects.
- Some information, such as the current contents of the Open Media interaction, current notepad, and so on, exist only within the browser and are accessible from the AIL only when a save was performed.

Chapter

4 Navigation and Controls

This chapter outlines the customization that enables:

- Keyboard navigation in custom pages
- Genesys Desktop shortcuts in Genesys Desktop 7.6
- Messaging functions for the Status Bar

See *Genesys Desktop 7.6 XML Tag Reference* for detailed descriptions of the tags related to the Keyboard Navigation API.

This chapter contains the following sections:

Customization Principles

This feature provides a keyboard navigation API that enables you to integrate custom controls in the Genesys Desktop mechanism. It also enables you to use Genesys Desktop shortcuts and custom shortcuts in your custom panels.

This feature is available for Agent Desktop Customization and in Open Media Extensions.

includeGDFunctionalities Function

The `includeGDFunctionalities` JavaScript function enables you to deploy certain JavaScript functions in your custom Genesys Desktop pages.

The `includeGDFunctionalities` function is fully described as:

```
window.parent.includeGDFunctionalities(window)
```

Call this function in the header of your custom page, if your custom page is displayed in a popup window, by using `window.parent` or `window.opener`.

The `includeGDFunctionalities` function takes the following parameter:

- `win`—The `window` object of the current custom page.

The `includeGDFunctionalities` function is available in the Genesys Desktop Customization API and Extension API for Tags and child tags. It can be accessed by sub frame (`window.parent`) or popup window (`window.opener`). Refer to Genesys Desktop 7.6 XML Tag Reference for information about the availability of this function for specific Tags and child tags.

Generated JavaScript Functions

The `includeGDFunctionalities` function activates Genesys Desktop shortcuts and creates some JavaScript functions in your custom page. The following is a snippet of a custom page that uses this function to provide other custom function to this page:

```
<html>
    <head>
        <script>
            window.parent.includeGDFunctionalities(window);
           ...
        <script>
    </head>
    <body>
        ...
    </body>
</html>
```

The following functions become available in the custom page after they are generated by the `includeGDFunctionalities` function:

- `addControl(control,nextControl)`—Add the navigable list control.

  Parameters:

  - `control`—A JavaScript object that implements the methods described below.
  - `nextControl`—Inserts a control before another control. If this parameter is not specified, the control is added to the end of the list.

- `createControlFromFrame(name)`—Creates an object control for FRAME and IFRAME. It is used to navigate in subpages.

  Parameter:

  - `name`—Specifies the name of the FRAME or IFRAME that is present in the current document.

  Returns:

  - JavaScript Object that provides the method:
    - `setFocusable`—Specifies if the control can receive the focus.

    Parameter:

- `focusable`—Boolean values: `true` or `false`.

Note: By default this returned control is focusable.

- `focusOnWindow()`—Set the focus on the window, this is used for full JavaScript control.
- `includeGDFunctionalities(window)`—A function that is used by frames and popup windows to propagate functions in subareas (FRAME, IFRAME, and popup window). See "includeGDFunctionalities Function" on page 95.
- `showWarningMessage(message)`—Display a warning message in the status bar.

  Parameter:

  - `message`—The message to display in the status bar.
- `showInformationMessage(message)`—Display an information message in the status bar.

  Parameter:

  - `message`—The message to display in the status bar.

Snippet Demonstrating Usage of a Custom Panel

The following is a snippet of a custom panel from the customer records area that is displayed in a tab. This customization activates the keyboard navigation and shortcut features in the current page and puts an internal IFRAME in the system.

1. Create the customization file `custom/custom.xml`:

```
<gcn-resources>
    <desktop>
        <customer-records>
            <tabs>
                <tab name="additionalInformation">
                    <text>Additional Information</text>
                    <javascript-onselect>
                        <![CDATA[
                            var form = document.forms.additionalInformationForm;
                            form.target = getDetailFrameName();
                            form.elements.idInteraction.value = idInteraction;
                            form.elements.idContact.value = idContact;
                            form.submit();
                        ]]>
                    </javascript-onselect>
                    <html-body>
                        <![CDATA[
                            <form name="additionalInformationForm"
   action="custom/additionalInformation.jsp">
                            </form>
                        ]]>
```

```
                </html-body>
            </tab>
        </tabs>
    </customer-records>
</desktop>
</gcn-resources>
```

2. Create a DHTML page, `custom/additionalInformation.jsp`:

```
<html>
    <head>
        <script language="javascript">
            window.parent.includeGDFunctionalities(window);
            function load() {
                activateGDFunctionalities();
            }
            function unload() {
                unactivateGDFunctionalities();
            }
           ...
        <script>
    </head>
    <body onload="load()" onunload="unload()">
        ...
        <iframe name="createControlFromFrame"
          src="subAdditionalInformation.jsp></iframe>
        ...
        <script>
            var subFrame = createControlFromFrame("createControlFromFrame");
            ...
            addControl(subFrame);
            ...
        </script>
    </body>
</html>
```

Create a DHTML page, `custom/subadditionalInformation.jsp`:

```
<html>
    <head>
        <script language="javascript">
            window.parent.includeGDFunctionalities(window);
            function load() {
                activateGDFunctionalities();
            }
            function unload() {
                unactivateGDFunctionalities();
            }
           .
        <script>
    </head>
    <body onload="load()" onunload="unload()">
        .
```

```
    </body>
</html>
```

Custom Control Interface For Keyboard Navigation

The customization code features a control object.

The following are the mandatory methods for the control object:

- `isFocusable`—Specifies whether the control can receive the focus. This method returns `false` if the control is hidden or disabled. The method is used by the system to set or remove the focus on the control object.
- `focus`—The method that is called by Genesys Desktop to apply the focus on the control object.

The following are optional methods for the control object:

- `onArrowLeft`—Called by the system if the left arrow is pressed.
- `onArrowRight`—Called by the system if the right arrow is pressed.
- `onArrowUp`—Called by the system if the up arrow is pressed.
- `onArrowDown`—Called by the system if the down arrow is pressed.
- `onEnter`—Called by the system if `Enter` is pressed.
- `onSpace`—Called by the system if the Space bar is pressed.
- `onEscape`—Called by the system if `Esc` is pressed.

Existing Customizations and Extensions that Use Custom Control

The following Genesys Desktop customization samples employ the `includeGDFunctionalities` JavaScript function:

- `sample1 - desktop`
- `sample3 - button-bar`
- `sample4- customer-records`
- `sample5 - toolbar`
- `sample6 - interaction-information`
- `sample7 - contact-ext`
- `sample9 - central-panel`
- `sample10 - client-notification`
- `sample11 - scheduling client notification`
- `sample12 - external knowledgebase`
- `sample13 - custom area`
- `sample14 - optimized size`

The following Genesys Desktop extension is employ the `includeGDFunctionalities` JavaScript function:

- `sms-out`

htmlcontrol.js Several Genesys Desktop customization samples use `htmlcontrol.js`. It contains some samples of custom control that implement the specific interface to be included in the keyboard navigation mechanism of Genesys Desktop. The controls are:

- `Input`
- `button image`
- `check box`
- `toggle button`
- `radio`

The following Genesys Desktop customization samples employ the `htmlcontrol.js` function:

- `sample6 - interaction-information`
- `sample7 - contact-ext`
- `sample11 - scheduling client notification`
- `sample12 - external knowledgebase`
- `sample13 - custom area`
- `sms-out`

The following is the `htmlcontrol.js` code:

```
function custom_HTMLInputControl(id,win) {
  this.id = id;
  this.win = win;
  this.focused = false;
  this.enabled = true;

  this.activate();
};

custom_HTMLInputControl.prototype = {

  activate:function() {
    this.getControl();
    this.control.onfocus =
custom_createCallback(this,this.onFocus,[]);
    this.control.onblur =
custom_createCallback(this,this.onUnfocus,[]);
  },

  useDisabled:function() {
    var controlTagName = this.control.tagName.toLowerCase();
    if (controlTagName == "select")
      return true;
    return false;
```

```
},

getControl:function() {
 if (this.control != null)
    return this.control;
  this.control = this.win.document.getElementById(this.id);
  if (this.useDisabled()) {
    if (this.control.readOnly) {
      this.control.disabled = true;
      this.control.readOnly = false;
    }
    if (this.control.disabled)
      this.disable();
    else
      this.enable();
  } else {
    if (this.control.disabled) {
      this.control.disabled = false;
      this.control.readOnly = true;
    }
    if (this.control.readOnly)
      this.disable();
    else
      this.enable();
  }
  return this.control;
},

focus:function() {
 this.getControl();
  if ( (!this.focused) && (this.isFocusable()) ) {
    try {
      this.control.focus();
      this.onFocus();
    } catch(e) {
    }
  }
},

onFocus:function() {
  this.focused = true;
  focusOnControl(this);
},

unfocus:function() {
},

onUnfocus:function() {
  unfocusOnControl(this);
  this.focused = false;
},
```

```
  isFocusable:function() {
   this.getControl();

    if (custom_isJsObjectHidden(this.control))
      return false;

    if (this.useDisabled())
      return !this.control.disabled;

    return !this.control.readOnly;
  },

  enable:function() {
   this.getControl();
    this.enabled = true;
    if (this.control != null) {
      if (this.useDisabled())
        this.control.disabled = false;
      else
        this.control.readOnly = false;
    }
  },

  disable:function() {
   this.getControl();
    this.enabled = false;
    if (this.control != null) {
      if (this.useDisabled())
        this.control.disabled = true;
      else
        this.control.readOnly = true;
    }
  },

  getValue:function() {
   this.getControl();
    if (this.control != null)
      return this.control.value;

    return "";
  },

  setValue:function(value) {
   this.getControl();
    if (this.control != null)
      this.control.value = value;
  }
}
```

```
function custom_HTMLButtonImage(id, text, imagesPath, imageHeight,
callbackFunction, positionControl, win) {
  this.id = id;
  this.text = text;
  this.tooltip = "";
  this.imagesPath = imagesPath;
  this.positionControl = positionControl;
  this.callbackFunction = callbackFunction;
  this.hrefCallbackFunction = "javascript:try {" +
this.callbackFunction + ";custom_nothingHTML();} catch(e) {}";
  this.win = win;
  this.w = -1;
  this.h = imageHeight;
  this.depth = 0;
  this.index = 0;
  this.imageTypeNumber = 1
  this.enabled = true;
  this.focused = false;
  this.shown = true;
  this.elm = null;

}

custom_HTMLButtonImage.prototype = {

  activate:function() {
    this.elm = this.win.document.getElementById(this.id);
    this.a = this.win.document.createElement("A");
    this.a.href = "javascript:";
    this.a.onfocus = custom_createCallback(this,this.focus,[]);
    this.elm.appendChild(this.a);
    for (var inc = 0;inc < this.imagesPath.length;inc++) {
      var img = this.win.document.createElement("IMG");

      if (inc == 0) {
        this.image = img
        this.image.title = this.tooltip;
        if (this.w != -1) {
          this.image.width = this.w;
          this.image.height = this.h;
        }
        img.id = this.id + "Img";
        img.border = 0;
        this.a.appendChild(img);
      }
      img.src = this.imagesPath[inc];
    }
    this.image.alt = this.text;

    if (this.enabled) {
      this.enable();
```

```
      if (this.focused)
        this.focus();
    } else
      this.disable();
  },

  disable:function() {
    this.enabled = false;
    this.index = 1;
    if (this.a == null)
      return;
    this.changeImage();
    this.a.href = "javascript:";
    this.a.tabIndex = -1;
  },

  enable:function() {
    this.enabled = true;
    if(this.focused)
      this.index = 2;
    else
      this.index = 0;
    if (this.a == null)
      return;
    this.changeImage();
    this.a.href = this.hrefCallbackFunction;
    this.a.tabIndex = 0;
  },

  focus:function() {
    if(this.enabled && !this.focused) {
      this.focused = true;
      this.index = 2;
      if (this.a == null)
         return;
      this.changeImage();
      this.win.focus()
      this.a.focus();

focusOnControl(this.win.custom_hTMLButtonImages[this.positionContro
l]);
    }
    return true;
  },

  unfocus:function() {
    this.focused = false;
    if (this.enabled)
      this.enable();
    else
      this.disable();
```

```
unfocusOnControl(this.win.custom_hTMLButtonImages[this.positionCont
rol]);
  },

  isFocusable:function() {
    return this.enabled && this.shown;
  },

  setDepth:function(depth) {
    this.depth = depth;
    if (this.image == null)
      return;
    this.changeImage();
  },

  changeImage:function() {
    var theIndex = this.index + this.depth;
    if (theIndex < this.imagesPath.length)
      this.image.src = this.imagesPath[theIndex];
  },

  setTooltip:function(text) {
    this.tooltip = text;
    if (this.image != null)
       this.image.title = this.tooltip;
  },

  show:function() {
    this.shown = true;
    this.win.custom_showJsObjectById(this.id);
  },

  hide:function() {
    this.shown = false;
    this.win.custom_hideJsObjectById(this.id);
  },

  setImages:function(imagesPath) {
   this.imagesPath = imagesPath;
   this.image.src = this.imagesPath[this.index];
  },

  onEnter:function(){
  }
}

function custom_HTMLCheckBox(id, text, checked, callbackFunction,
positionControl, win) {

  this.checked = checked;
```

```
  this.imageTypeNumber = 2;
  this.callbackFunction = callbackFunction;

  this.hTMLButtonImage = new
custom_HTMLButtonImage(id,text,["checkboxcheck.gif",
"checkboxcheckdisabled.gif",  "checkboxcheckfocused.gif",

"checkboxuncheck.gif","checkboxuncheckdisabled.gif","checkboxunchec
kfocused.gif"],
                                                        15,
callbackFunction, positionControl, win);
  this.hTMLButtonImage.hrefCallbackFunction =
"javascript:custom_clickHTMLControl(" + positionControl + ");" +
this.callbackFunction;
  this.manageDisplay();

  this.activate();
}

custom_HTMLCheckBox.prototype = {

  activate:function() {
      this.hTMLButtonImage.activate();
  },

  enable:function() {
      this.hTMLButtonImage.enable();
  },

  disable:function() {
      this.hTMLButtonImage.disable();
  },

  focus:function() {
      this.hTMLButtonImage.focus();
  },

  unfocus:function() {
      this.hTMLButtonImage.unfocus();
  },

  isFocusable:function() {
      return this.hTMLButtonImage.isFocusable();
  },

  show:function() {
      this.hTMLButtonImage.show();
  },

  hide:function() {
      this.hTMLButtonImage.hide();
```

```
  },

  isCheck:function(){
      return this.checked;
  },

  setCheck:function(checked) {
    this.checked = checked;
    this.manageDisplay();
  },

  click:function() {
    this.setCheck(!this.checked);
  },

  manageDisplay:function() {
    if (this.checked)
      this.hTMLButtonImage.setDepth(0);
    else
      this.hTMLButtonImage.setDepth(3);
  },

  onEnter:function() {
   this.hTMLButtonImage.onEnter();
  },

  onSpace:function(){
  },

  onArrowLeft:function(){
  },

  onArrowRight:function(){
  },

  onArrowUp:function(){
  },

  onArrowDown:function(){
  }
}

function
custom_HTMLToggleButton(id,text,imagesPath,height,checked,callbackF
unction,positionControl,win) {
  this.hTMLButtonImage = new
custom_HTMLButtonImage(id,text,imagesPath,height,callbackFunction,
positionControl, win);

  this.checked = checked;
  this.imageTypeNumber = 2;
```

```
  this.callbackFunction = callbackFunction;
  this.hTMLButtonImage.hrefCallbackFunction =
"javascript:custom_clickHTMLControl(" + this.positionControl + ");"
+ this.callbackFunction;
  this.manageDisplay();
}

custom_HTMLToggleButton.prototype = {

  activate:function() {
      this.hTMLButtonImage.activate();
  },

  enable:function() {
      this.hTMLButtonImage.enable();
  },

  disable:function() {
      this.hTMLButtonImage.disable();
  },

  focus:function() {
      this.hTMLButtonImage.focus();
  },

  unfocus:function() {
      this.hTMLButtonImage.unfocus();
  },

  isFocusable:function() {
      return this.hTMLButtonImage.isFocusable();
  },

  show:function() {
      this.hTMLButtonImage.show();
  },

  hide:function() {
      this.hTMLButtonImage.hide();
  },

  setCheck:function(checked) {
    this.checked = checked;
    this.manageDisplay();
  },

  click:function() {
    this.setCheck(!this.checked);
  },

  setTooltip:function(text) {
```

```
    this.hTMLButtonImage.setTooltip(text);
  },

  manageDisplay:function() {
    if (this.checked)
      this.hTMLButtonImage.setDepth(0);
    else
      this.hTMLButtonImage.setDepth(3);
  }
}

function
custom_HTMLRadio(id,text,selected,groupName,callbackFunction,positi
onControl,win) {
  this.hTMLButtonImage = new
custom_HTMLButtonImage(id,text,["radioselect.gif",

"radioselectdisabled.gif",

"radioselectfocused.gif",

"radiounselect.gif",

"radiounselectdisabled.gif",

"radiounselectfocused.gif"],
                                                    12,
callbackFunction,positionControl,win);
  this.groupName = groupName;
  this.imageTypeNumber = 2;
  this.checked = selected;
  this.callbackFunction = callbackFunction;
  this.hTMLButtonImage.hrefCallbackFunction =
"javascript:custom_clickHTMLControl(" + this.positionControl + ");"
+ this.callbackFunction;
  this.manageDisplay();
}

custom_HTMLRadio.prototype = {
  activate:function() {
      this.hTMLButtonImage.activate();
  },

  enable:function() {
      this.hTMLButtonImage.enable();
  },

  disable:function() {
      this.hTMLButtonImage.disable();
  },
```

```
  focus:function() {
      this.hTMLButtonImage.focus();
  },

  unfocus:function() {
      this.hTMLButtonImage.unfocus();
  },

  isFocusable:function() {
      return this.hTMLButtonImage.isFocusable();
  },

  show:function() {
      this.hTMLButtonImage.show();
  },

  hide:function() {
      this.hTMLButtonImage.hide();
  },

  setCheck:function(checked) {
    this.checked = checked;
    this.manageDisplay();
  },

  click:function() {
    this.select();
  },

  select:function() {
    if (this.checked == true)
      return;
      for(var inc = 0;inc <
this.hTMLButtonImage.win.custom_hTMLButtonImages.length;inc++) {
        if (this.groupName ==
this.hTMLButtonImage.win.custom_hTMLButtonImages[inc].groupName) {
          if (this.hTMLButtonImage.win.custom_hTMLButtonImages[inc]
== this)

this.hTMLButtonImage.win.custom_hTMLButtonImages[inc].checked =
true;
          else

this.hTMLButtonImage.win.custom_hTMLButtonImages[inc].checked =
false;

this.hTMLButtonImage.win.custom_hTMLButtonImages[inc].manageDisplay
();
        }
      }
  },
```

```
  manageDisplay:function() {
    if (this.checked)
      this.hTMLButtonImage.setDepth(0);
    else
      this.hTMLButtonImage.setDepth(3);
  }
}


  custom_controls = [];
  custom_createHTMLInputControl = function(id) {
    var b = new custom_HTMLInputControl(id,window);
    custom_controls[custom_controls.length] = b;

    return b;
  }

  custom_hTMLButtonImages = [];
  custom_createHTMLButtonImage = function(id, text, imagePath,
imageHeight, callbackFunction) {
    var b = new custom_HTMLButtonImage(id, text, imagePath,
imageHeight, callbackFunction, custom_hTMLButtonImages.length,
window);

    custom_hTMLButtonImages[custom_hTMLButtonImages.length] = b;
    custom_controls[custom_controls.length] = b;
    b.activate();

    return b;
  }

  custom_createHTMLCheckBox =
function(id,text,checked,callbackFunction) {
    var b = new custom_HTMLCheckBox(id, text, checked,
callbackFunction, custom_hTMLButtonImages.length, window);

    custom_hTMLButtonImages[custom_hTMLButtonImages.length] = b;
    custom_controls[custom_controls.length] = b;

    return b;
  }

  custom_createHTMLToggleButton =
function(id,text,imagePath,height,checked,callbackFunction) {
    var b = new
custom_HTMLToggleButton(id,text,imagePath,height,checked,callbackFu
nction,custom_hTMLButtonImages.length,window);

    custom_hTMLButtonImages[custom_hTMLButtonImages.length] = b;
    custom_controls[custom_controls.length] = b;
```

```
    return b;
  }

  custom_createHTMLRadio =
function(id,text,selected,groupName,callbackFunction) {
    var b = new
custom_HTMLRadio(id,text,selected,groupName,callbackFunction,custom
_hTMLButtonImages.length,window);

    custom_hTMLButtonImages[custom_hTMLButtonImages.length] = b;
    custom_controls[custom_controls.length] = b;

    return b;
  }

  custom_clickHTMLControl = function(positionControl){
    custom_hTMLButtonImages[positionControl].click();
  }

  custom_nothingHTML = function() {
      return;
  }

  custom_createCallback = function(object,func,parameters) {
    return function() {
      var param = arguments;

      if (parameters != null)
        param = parameters;

      return func.apply(object,param);
    }
  }

  custom_showJsObject = function(object) {
     object.style.visibility = "visible";
     object.style.display = "block";
  }

  custom_hideJsObject = function(object) {
     object.style.visibility = "hidden";
     object.style.display = "none";
  }

  custom_showJsObjectById = function(id) {
    var object = document.getElementById(id);
    if (object != null)
      custom_showJsObject(object);
  }
```

```
  custom_hideJsObjectById = function(id) {
    var object = document.getElementById(id);
    if (object != null)
      custom_hideJsObject(object);
  }

  custom_isJsObjectHidden = function(object) {
    if ( (object.style != null) && (
          ( (object.style.display != null) && (object.style.display
== "none") )
          || ( (object.style.visibility != null) &&
(object.style.visibility == "hidden") ) ) )
      return true;
    return custom_isObjectHidden(object);
  }

  custom_isObjectVisible = function(object) {
    return !(custom_doObjectClassNameContainsClass(object,
"hiddenObj"));
  }

  custom_isObjectHidden = function(object) {
    return custom_doObjectClassNameContainsClass(object,
"hiddenObj");
  }

  custom_doObjectClassNameContainsClass = function(object, cssClass)
{
    if (object.className == null)
      return false;
    if (object.className.indexOf(cssClass) == -1)
      return false;
    return true;
  }
```

Internal Objects, Functions, and Classes Generated in Custom Pages

The following Internal Objects, Functions, and Classes are deployed for Custom Pages.

Note: Ensure that your custom code does not conflict with the internal classes, objects, constants, and functions.

Internal Classes

- Vector
- Listener
- Message
- Bus
- KeyboardManager
- FocusManager

Internal Objects

- browser
- bus
- keyboardManager
- focusManager

Internal Constants

- BROWSER_TYPE_IE
- BROWSER_TYPE_SAFARI
- BROWSER_TYPE_NETSCAPE
- DISPATCH_MODE_ALL
- DISPATCH_MODE_CHILD_EXCLUDE
- DISPATCH_MODE_PARENT_EXCLUDE
- DISPATCH_MODE_PARENT_ONLY
- DISPATCH_MODE_DEPTH
- DISPATCH_MODE_PARENT_DEPTH
- DISPATCH_MODE_CHILD_DEPTH
- KEY_ENTER
- KEY_SPACE
- KEY_ESC
- KEY_ARROW_UP
- KEY_ARROW_DOWN
- KEY_ARROW_LEFT
- KEY_ARROW_RIGHT
- KEY_BACKSPACE
- KEY_B
- KEY_C
- KEY_D
- KEY_E
- KEY_I
- KEY_V
- KEY_X

- KEY_A
- KEY_F
- KEY_H
- KEY_L
- KEY_O
- KEY_P
- KEY_R
- KEY_U
- KEY_Y
- KEY_N
- KEY_Z
- KEY_T
- KEY_K
- KEY_ADD
- KEY_SUB
- KEY_0
- KEY_TAB
- DIRECTION_PREVIOUS
- DIRECTION_NEXT

Internal Functions

- setFocusOnObject
- createCallback
- createGlobalVariable
- deleteGlobalVariable
- dynamicCallMethod

Limitations

The includeGDFunctionalities JavaScript function has the following limitations:

- Customizations must be present on the same host as Genesys Desktop. This is due to browser security constraints that prevent access to information on another domain.
- Customized keyboard navigation works only for pages and controls that implement the Keyboard Navigation API.
- Customized keyboard navigation only works in a child custom-frame if the parent custom-frame implements it.
- The keyboard navigation feature is not compatible with external accessibility tools such as the Freedom Scientific application: Job Access With Speech (JAWS). Use standard HTML controls to support this type of third party application.

Navigation and Control API Example

There is no specific example for the Navigation and Control functionalities. All the Genesys Desktop customization samples employ the navigation and controls functionalities; however, the following customization samples focus on keyboard navigation:

- `sample6 - interaction-information`
- `sample7 - contact-ext`
- `sample9 - central panel`
- `sample11 - scheduling client notification`
- `sample12 - external knowledgebase`
- `sample13 - custom area`
- `open-media-extension\extension\sms-out`

The other samples employ the navigation and controls functionalities for shortcuts only, or to publish a message in the status bar.

Chapter

5 Localization in JSP and Java Code

This chapter outlines the customization that enables you to localize from JSP and Java code.

See *Genesys Desktop 7.6 XML Tag Reference* for detailed descriptions of the localization in Java code and JSP.

This chapter contains the following sections:

Customization Principles

This feature provides a mechanism to localize custom page content. Use this method to localize labels and image buttons from customization. This feature is based on the dictionary (for detailed information, refer to"Dictionary Files" on page 27).

Using Tag Library

You can localize labels and button images from the dictionnary by using the Tag Library that is provided with the Genesys Desktop. The Tag Library is described in the following file: `WEB-INF/custom.tld file`

To use this Tag Library in a custom JSP file, add the following code in the header:

```
<%@ taglib uri="custom" prefix="custom" %>
```

The following custom tags are available:

- `custom:button`—Generates an image button that has the Genesys Desktop look and feel. It also generates a string that has the following format:

```
["<path to the button image normal state>",  "<path to the button
image disabled>",  "<path to the button image with the focus>"].
```

  This string is directly usable by the common JavaScript library that is delivered with the Genesys Desktop Customization samples.
- `custom:messageHTML`—Generates a label to be included in an HTML part.
- `custom:messageJavaScript`—Generates a label to be included in a JavaScript part.

Using Java code

You can obtain localized strings from the dictionary by using Java code. To get this information from your JSP or servlet, you should employ an instance of the `com.genesyslab.uadthin.customization.gui.CustomLocalization` interface that is provided by the `customLocalization` attribute of the `javax.servlet.http.HttpSession` object. The instance is added at the login step. Use the following public methods:

- `public String getDictionaryMessage(String dictionaryClass, String dictionaryKey);`
- `public String getDictionaryMessage(String dictionaryClass, String dictionaryKey, String arg);`
- `public String getDictionaryMessage(String dictionaryClass, String dictionaryKey, String arg0, String arg1);`
- `public String getDictionaryMessage(String dictionaryClass, String dictionaryKey, String arg0, String arg1, String arg2);`
- `public String getDictionaryMessage(String dictionaryClass, String dictionaryKey, String arg0, String arg1, String arg2, String arg3);`
- `public String getDictionaryMessage(String dictionaryClass, String dictionaryKey, String arg0, String arg1, String arg2, String arg3, String arg4);`

Note: This interface is included in the following Genesys Desktop customization `.jar` file:

`WEB-INF/lib/gdesktop-customization-api.jar`

You can use this in your development environment to compile your java file.

Use the following code snippet as an example of how to localize your code:

1. Import the `CustomLocalization` class:

```
<%@ page
import="com.genesyslab.uadthin.customization.gui.CustomLocalization
" %>
```

2. Retrieve localization information from the session:

```
<%
    CustomLocalizion customLocalization = (CustomLocalization)
session.getAttribute("customLocalization");

%>
```

3. Apply the localization this way:

```
<%= customLocalization.getDictionaryMessage("custom", "message") %>
```

Localization Demonstration Snippet

The code snippet provides only one example of usage for one method; however, you can use others from “Using Java code” on page 118.

```
<%@ page language="java" buffer="none" contentType="text/html;
charset=utf-8" %>
<%@ page
import="com.genesyslab.uadthin.customization.gui.CustomLocalization
" %>
<%@ taglib uri="custom" prefix="custom" %>

<%
    CustomLocalization customLocalization = (CustomLocalization)
session.getAttribute("customLocalization");

%>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-
8">
    <link rel="stylesheet" type="text/css" href="style.css"/>
    ...
  </head>
    <body class="fontName backgroundPanel" leftmargin="0"
topmargin="0" marginwidth="0" marginheight="0"
onload="custom_load()" onunload="custom_unload()">
   <table width="100%" border="0" cellpadding="0" cellspacing="0">
      <tr valign="top">
        <td class="backgroundPanel" align="left">
          <table class="editablePanelText" id="knowledgeTable"
name="knowledgeTable" width="100%" border="0" cellspacing="6"
cellpadding="6">
            <tr height="16" id="row0" class="tableBackgroundItem">
```

```
           <td width="20%" id ="row0_0">:-)</td>
               <td width="70%" id="row0_1"><custom:messageHTML
dictionaryClass="custom" dictionaryKey="happy" /></td>
            </tr>
            <tr height="16" id="row1" class="tableBackgroundItem">
           <td width="20%" id ="row1_0">:]-[</td>
               <td width="70%" id="row1_1"><custom:messageHTML
dictionaryClass="custom" dictionaryKey="angry" /></td>
            </tr>
            <tr height="16" id="row2" class="tableBackgroundItem">
           <td width="20%" id ="row2_0">:-(</td>
               <td width="70%" id="row2_1" ><%=
customLocalization.getDictionaryMessage("custom", "sad") %></td>
            </tr>
          </table>
        </td>
      </tr>
      <tr><td height="5" colspan="2"><img src="empty.gif"></td></tr>
      <tr align="center">
       <td>
       <div id="insertDiv"></div>
       </td>
      </tr>
    </table>
    ...
  </body>
</html>
```

Localization in Java code and JSP Example

Genesys Desktop does not inlcude a specific example for the Localization functionality; however, you can use the following customization samples to test your localization code:

- sample6 - interaction-information
- sample7 - contact-ext
- sample9 - central panel
- sample11 - scheduling client notification
- sample12 - external knowledgebase
- sample13 - custom area
- open-media-extension\extension\generic
- open-media-extension\extension\fax
- open-media-extension\extension\sms-in
- open-media-extension\extension\sms-out

Chapter

6 Agent and Supervisor Desktop Interception API

This chapter outlines the customization that enables server-side interception of agent and supervisor actions in Genesys Desktop 7.6. See *Genesys Desktop 7.6 XML Tag Reference* for detailed descriptions of the tags related to the Interceptor API.

This chapter contains the following sections:

Customization Principles

The interception mechanism enables you to create a trigger to execute custom Java code in specific situations, such as agent actions and some system events, before Genesys Desktop Server executes its own code.

Interception API relies on Genesys Desktop actions that relate to the context where a particular action of interest is executed, such as `Agent` (the one executing the action to be traced), `Contact`, `Interaction`, and so on, along with J2EE interfaces, such as `Session`. Table 2 on page 122 contains a complete list of the actions that you can trace.

Note: You must write one implementation class of the Configuration, Code Example, and Deployment example for each of the actions that you wish to intercept.

Table 2: List of traceable actions

Action name	Action	Out-of-the-box Property Name	Implementation Interface	Scope[a]
VIEW_CONTACT	Agent selects a contact in a list or within an active interaction to view it.	agent	com.genesyslab.ail.Agent	IN
		contact	com.genesyslab.ail.Contact	IN
		interaction	com.genesyslab.ail.Interaction	IN
VIEW_HISTORY_INTERACTION	Agent selects an interaction in a contact history to view it.	agent	com.genesyslab.ail.Agent	IN
		contact	com.genesyslab.ail.Contact	IN
		interaction	com.genesyslab.ail.Interaction	IN
SERVER_START	Genesys Desktop server is initializing.			IN
SERVER_STOP	Genesys Desktop server is stopping.			IN
SESSION_START	Agent starts a session.	agent	com.genesyslab.ail.Agent	IN
		session	javax.servlet.http.HttpSession	IN
SESSION_STOP	Agent ends her or his session.	agent	com.genesyslab.ail.Agent	IN
PLACE_IN_WORKBIN	Agent places an interaction in an workbin.	agent	com.genesyslab.ail.Agent	IN
		interactionMultimedia	com.genesyslab.ail. InteractionMultimedia	IN
		workbinTarget	com.genesyslab.ail.workflow. Workbin	IN

Table 2: List of traceable actions (Continued)

Action name	Action	Out-of-the-box Property Name	Implementation Interface	Scope[a]
MOVE_TO_WORKBIN	Agents moves an interaction to a workbin.	agent	com.genesyslab.ail.Agent	IN
		idInteraction	java.lang.String	IN
		attachedData	java.util.Map	IN/OUT
		idWorkbinSource	java.lang.String	IN
		workbinTarget	com.genesyslab.ail.workflow.Workbin	IN
SUPERVISOR_MOVE_TO_WORKBIN	Configured as Genesys Supervisor Desktop interceptor. It is executed before the standard Supervisor Move to Workbin action.	agent	com.genesyslab.ail.Agent	IN
		interactionMultimedia	com.genesyslab.ail.InteractionMultimedia	IN
		idWorkbinSource	java.lang.String	IN
		workbinTarget	com.genesyslab.ail.workflow	IN
ALARM	Reports Genesys Supervisor Desktop alerts and other events.	resourceDBID	java.lang.Integer	IN
		resourceName	java.lang.String	IN
		resourceType	java.lang.Integer	IN
		dataItemID	java.lang.String	IN
		level	java.lang.Integer	IN

Table 2: List of traceable actions (Continued)

Action name	Action	Out-of-the-box Property Name	Implementation Interface	Scope[a]
SKILL_AUDIT	Activated by a click on the Save button in the Properties tab. It is applicable to the update of only one agent. Or, activted by the selection of the Assign or Remove menu item from the Actions menu. It is applicable for the simulatneous update of several agents	supervisor	com.genesyslab.ail.Agent	IN
		agentUserName	String	IN
		agentEmployeeId	String	IN
		agentFirstName	String	IN
		agentLastName	String	IN
		agentDBID	String	IN
		skills	com.genesyslab.ail.Skill []	IN
		actions	int []	IN
		reason	String	IN

a. IN: the property is filled by Genesys Desktop.The Property should have a setter. OUT: the property is filled by the custom bean and used by Genesys Desktop after the intercepted part of the code is executed. The property should have a getter.

Table 3 describes the interceptor tags and attributes that you can use in your configuration file.

For each property declared in the configuration file, you must write a `set` java method that uses the regular Java Bean approach. For example, the property `interaction` is translated into the Java methods: `setInteraction`.

Table 3: Description of the bean tags and attributes

Tag	Attribute	Description
interceptor-config		This is the topmost tag. Every Interceptor customization file must start with this tag. It encloses all tags that define the actions to be intercepted.
interceptors		This tag must follow the <interceptor-config> tag and contain all the individual custom interceptors that you create. The individual custom interceptors are contained by the <interceptor> tag.

Table 3: Description of the bean tags and attributes (Continued)

Tag	Attribute	Description
interceptor	name implementation-class method	You can create as many custom interceptors as you want. They must be contained by this tag. All the custom interceptors are grouped under the <interceptors> tag. This tag contains three required attributes: • **name**—the name for your custom interceptor. Must be named after one of the traceable actions listed in Table 2 on page 122. • **implementation-class**—the fully qualified class name of the traceable action. This class is written by your developer. • **method**—the method of your implementation class that you wish the interceptor framework to call when the associated action is triggered. Typically: **execute**.
property	name implementation-name type value	This is an optional tag that enables you to specify which properties of the traceable action you want to trace. Each property must be specified separately. The tag is closed by a "/". This tag includes one mandatory attribute and three optional attributes. • **name**—the name of the traceable property that you want to traces (see Table 2 on page 122 for Out-of-the-box property names). It can be part of the reserved property names for this interceptor or it can be a custom name. • **implementation-name**—(optional) the name of the property in your implementation class if it is different from the attribute name. This is compatible with the Java Bean definition of a property. • **type**—(optional) the type of the property defined in your implementation. Applicable only for custom properties. • **value**—(optional) the value to which your property will be set in the implemented object at runtime. Applicable only for custom properties.

The example in the section “Configuration, Code Example, and Deployment” on page 128 shows you how to execute some Java code that maintains an audit trail log in a plain text file.

Typical Interceptor Execution

This section describes how the Interceptor API functions in a typical agent action scenario (see Figure 18 on page 127):

1. The agent clicks a control in the browser, such as a button, link, or an item in a list.
2. The browser sends an HTTP request to Genesys Desktop Server.
3. The request is handled by the MVC framework within the Genesys Desktop web application.
4. The MVC Framework triggers the execution of the Java Action class configured and delivered in the out-of-the-box GAD.
   a. If there is a configured Interceptor Java class, this code is executed first.
   b. The out-of-the-box Java code associated to the action is then executed.

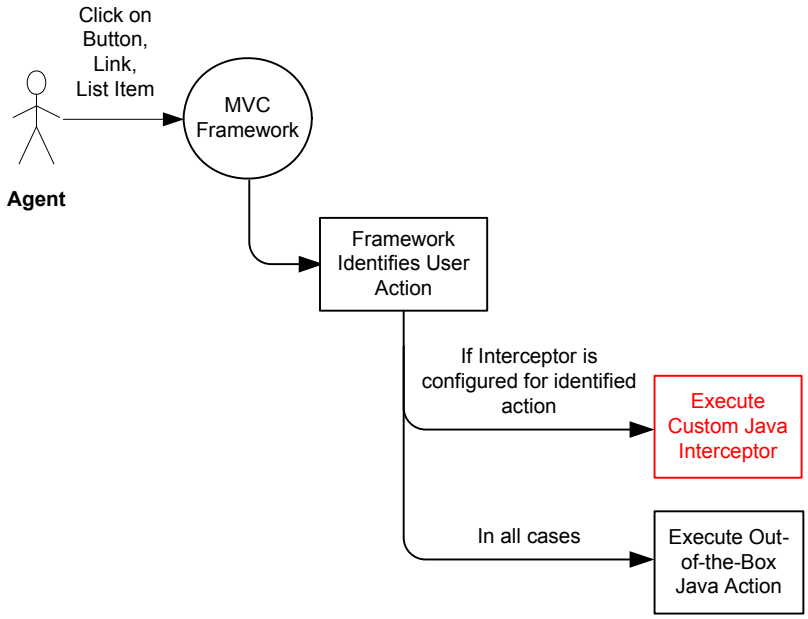


Figure 18: Execution of the custom interception code.

Deployment Recommendations

Use the following principles when deploying samples:

- The directory must be named: `interceptor`
- Copy at least one XML interceptor file to the `interceptor` directory. Each file must contain the full structure described in "Deployment" on page 131.
- Copy your compiled Java content to the appropriate directory:
  - if it is a `.jar` file, copy it to: `webapps/gdesktop/WEB-INF/lib`
  - if it is a set of `.class` files, copy them to: `webapps/gdesktop/WEB-INF/classes`

Note: Ensure that you maintain the structure of your package.

Interception API Example

This section presents a code example that illustrates possible ways to customize the Genesys Desktop 7.6 Interceptor. Genesys recommends that you carefully examine this example before you begin customizing Genesys Desktop. See *Genesys Desktop 7.6 XML Tag Reference* for detailed descriptions of the tags used in these examples.

Shipped Code Examples

The following example is shipped on Genesys Desktop DVD:

- `sample1 - Audit Trail`: Intercepting actions and storing them in a plain text file, as described in "Configuration, Code Example, and Deployment" on page 128.
- `sample2 - workbin-owner`: This example is not described in this chapter. Intercepting user actions that are related to workbin management. This sample adds a custom column in a Workbin to store the owner of the interaction.
- `sample3 - supervisor-alarm`: This example is not described in this chapter. Intercepting supervisor alarm messages and storing them in a plain text file.
- `sample4 - skill-audit-trail`: This example is not described in this chapter. Intercepting skill updates (adding, changing, and removing) and storing them in a plain text file.

The example in this section shows how to customize some of the functionality of the Genesys Desktop application.

Configuration, Code Example, and Deployment

This section presents an example configuration file, a code example, and deployment instructions. Use this example in conjunction with the code sample provided on the Genesys Desktop DVD.

Example Configuration File

```
<interceptor-config>
    <interceptors>
        <interceptor name="VIEW_CONTACT" implementation-
class="com.genesyslab.samples.interceptor.ViewContact"
method="execute">
            <property name="agent" implementation-name="myAgent"/>
            <property name="contact"/>
            <property name="interaction"/>
            <property name="customPropertyString"
type="java.lang.String" value="MY_CUSTOM"/>
```

```
        </interceptor>
        <interceptor name="VIEW_HISTORY_INTERACTION" implementation-
class="com.genesyslab.samples.interceptor.ViewHistoryInteraction"
method="execute">
            <property name="agent"/>
            <property name="contact"/>
            <property name="interaction"/>
        </interceptor>
        <interceptor name="SERVER_START" implementation-
class="com.genesyslab.samples.interceptor.StartServer"
method="execute">
            <property name="customRootPathName"
type="java.lang.String" value="../logs"/>
        </interceptor>
        <interceptor name="SERVER_STOP" implementation-
class="com.genesyslab.samples.interceptor.StopServer"
method="execute">
        </interceptor>
        <interceptor name="SESSION_START" implementation-
class="com.genesyslab.samples.interceptor.StartAgentSession"
method="execute">
            <property name="agent"/>
            <property name="session"/>
        </interceptor>
        <interceptor name="SESSION_STOP" implementation-
class="com.genesyslab.samples.interceptor.StopAgentSession"
method="execute">
            <property name="agent"/>
        </interceptor>
    </interceptors>
</interceptor-config>
```

Note: You must have one `<interceptor>` tag for each action that you want to intercept (see "Customization Principles" on page 121).

Example Code

The following is a code example that matches the configuration file shown in the "Example Configuration File" on page 128.

```
package com.genesyslab.samples.interceptor;

import com.genesyslab.ail.Agent;
import com.genesyslab.ail.Contact;
import com.genesyslab.ail.Interaction;

/**
 * This class is used to implement the behavior that must be executed when a user
   visualizes a contact record.
```

```
 * In this sample the behavior consists in calling a primitive of the AuditFileManager
   that writes
 * timestamp, contact ID, related interaction ID if any and user name in a plain text
   file.
 * In configuration file interceptor.xml it is configured in this XML fragment:
 *      <interceptor name="VIEW_CONTACT" implementation-
   class="com.genesyslab.samples.interceptor.ViewContact" method="execute">
 *          <property name="agent" implementation-name="myAgent"/>
 *          <property name="contact"/>
 *          <property name="interaction"/>
 *          <property name="customPropertyString" type="java.lang.String"
   value="MY_CUSTOM"/>
 *      </interceptor>
 */
public class ViewContact {
    private Agent myAgent;
    private Contact contact;
    private Interaction interaction;
    private String customPropertyString;

    /**
     * Called by interceptor framework prior to call to "execute" method.
     * This is a standard attribute.
     */
    public void setContact(Contact contact) {
        this.contact = contact;
    }

    /**
     * Called by interceptor framework prior to call to "execute" method.
     * This is a standard attribute.
     */
    public void setInteraction(Interaction interaction) {
        this.interaction = interaction;
    }

    /**
     * Called by interceptor framework prior to call to "execute" method.
     * This is a custom attribute used here as a fake
     */
    public void setCustomPropertyString(String customPropertyString) {
        this.customPropertyString = customPropertyString;
    }

    /**
     * Called by interceptor framework prior to call to "execute" method.
     * This is a standard attribute.
     * The corresponding property is declared in interceptor.xml has a
     * "implementation-name" attribute. So the name of this method is not based on
     * property name as for other properties, but on "implementation-name"
     */
```

```
    public void setMyAgent(Agent myAgent) {
        this.myAgent = myAgent;
    }

    /**
     * This method is referenced as attribute "method" in corresponding interceptor.xml
   tag.
     * It is called automatically by the framework after all the "in" properties have
   been set
     * thanks to the set method declared above.
     */
    public void execute() {
        AuditFileManager mgr = AuditFileManager.getInstance();
        if (contact!=null) {
            if (interaction==null) {
                mgr.append("Agent "+myAgent.getId()+" is accessing contact
   "+contact.getId());
            } else {
                mgr.append("Agent "+myAgent.getId()+" is accessing contact
   "+contact.getId()+" while handling interaction "+interaction.getId());
            }
        }
    }
}
```

Deployment

After you have written and compiled your classes, they must be deployed in the directory of the web application.

1. In the directory `webapps/gdesktop/`, create a directory named `interceptor/`.
2. Copy the file `interceptor.xml` into the `interceptor/` directory.
3. Copy the file `interceptor-audit-trail.jar` into the following directory:

   `webapps/gdesktop/WEB-INF/lib/`

The classes of your code corresponding to the interceptor implementation must be referenced in the configuration XML file (see “Example Configuration File”).

Chapter

7 Supervisor Desktop Customization

This chapter presents code examples that illustrate possible ways to customize the Genesys Supervisor Desktop 7.6 graphical user interface (GUI).

Customizing the Supervisor Desktop

There are three approaches to customizing the GUI of the Genesys Supervisor Desktop (GSD): wizards, XML files, and Web Services. Table 4 on page 134, summarizes the three approaches to customization, and defines where the customization is started, the scope of the customization (Supervisor vs. Application), and what is customized. Table 5 on page 134, summarizes how you use each customization approach.

Approaches to Customization

Through wizards, all data that have the scope of Supervisor are saved in the Configuration Layer, except for the default View Profile Template, which has scope of the entire application. Wizards are used to define what objects are shown and, in part, how they are shown. See "Customization Through Wizards" on page 134.

Through XML files, all data are stored on the server storage; the scope is the whole application. XML files define, in part, what data are shown and how they are shown, as well as what actions can be done with them. See "Customization Through XML Files" on page 135.

Web Services are described in "Third-Party Data Interface" on page 177. See also "Customization Through Web Services" on page 136.

Table 4: Approaches to Customizing Genesys Supervisor Desktop

Approach	Where Stored	Scope*	What Is Customized		
			Data	UI	Actions
Wizards	Configuration Server	Supervisor scope**	Yes	Yes	No
XML files	File storage (web container)	Application scope	Yes	Yes	Yes
Web services	Web service, file storage (web container)	Application scope	Yes	No	No

* with the exception of the default View Profile Template

** Supervisor scope means that customization items can be assigned to specific supervisors. Note: items are visible to all supervisors. Application scope means that customization items are used by all supervisors.

Table 5: How to Use the Three Approaches to Customization

Approach	How Accessed
Wizards	• Through the GUI • Through XML files in `gdesktop/supervisor/data` (default values for wizards)
XML files	• `gdesktop/WEB-INF/classes/com/genesyslab/uadthin/supervisor/resources/GSDResources.properties` (or localized file) • XML files in `gdesktop/supervisor/xml`
Web services	• `web.xml` • `userdata.xml` (how indicated in `web.xml`) • Web service (how indicated in `userdata.xml`)

Customization Through Wizards

Customization through wizards is described at length in the *Genesys Supervisor Desktop Help*, in the section: “Administration Tasks”. To summarize, there are three wizards that apply to the following levels:

- View Profile Template—consists of several object sets for different resource types). The default View Profile is used if you do not specify a View Profile Template.
- Object set—consists of a set of objects of a given resource type.
- View set—the internal structure for linking object sets and view templates.
- View Template—consists of statistics and properties, and their widths for one or several resource types.

For each of these levels, XML files define the default values in the following directory:

`supervisor/data`

Customization Through XML Files

This chapter focuses on customization through XML files. The following are the sets of XML files that you can use for customization:

- View set—consists of two sections: `list` and `detail`.
- Section—consists of several View Templates.
- View Template—consists of different UI elements.
- UI Elements—menus, actions, groups, property groups, child views, data items.

Note: View Templates are included in both wizards and XML files. They are similar in that they are represented in the GUI as tabs, but they are also different, in that customization through wizards enables you only to use the `statistics` and `properties` in tabular form; whereas, customization through XML files enables you to use tables as well as other types of GUI elements.

Another difference between wizards and XML files is that the View Templates that are saved in the Configuration Layer can be turned off through the wizards, while the other View Templates are stored in the XML files and so are always available, unless they are turned off by editing the XML code.

Customization through XML files is linked to resource types with the help of the following:

- `GSDResources.properties` file—defines view sets, titles, and images for all resource types which GSD can implement.
- `objectsets.xml` file—defines view sets, titles, and images for resource types that are definitely part of the object hierarchy. For example, skills are part of the hierarchy, but agents that are inside skills might or might not be a part of the hierarchy.

Customization Through Web Services

Customization through Web Services is described in the section "Third-Party Data Interface" on page 177.

Object Hierarchy/Hierarchy of UI Elements

Table 6 displays the hierarchy of UI elements, provides a key to the UI elements that are shown in Figure 19 on page 137, and indicates the approaches that can be used to customize the elements.

Table 6: Hierarchy of UI Elements for Customization in Genesys Supervisor Desktop

UI Element	Location in Figure 19	Wizards	XML Files	Web Services
View Profile Template	All elements in the drop-down list at area 1	Yes	Yes (default view profile only in `viewProfiles.xml`)	No
Object set	One folder element of area 1	Yes	Yes	No
View set	Areas 2 and 3	No	Yes	No
Section	Area 2 or area 3	No	Yes	No
View template	Each tab at are 4	Yes	Yes	No
Actions, menus, groups, property groups, child views, data items	Every element of the tabs at area 4	Yes (statistic data items only, through Statistic Wizard)	Yes	Yes (data items only)

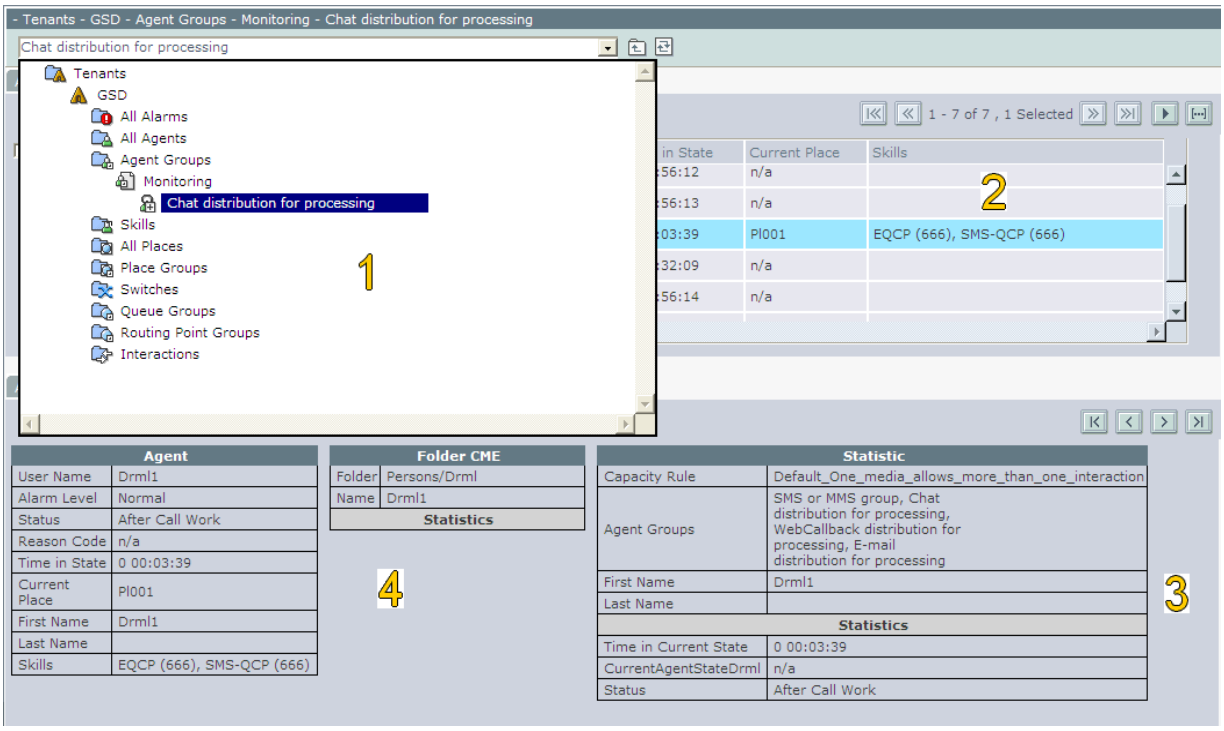


Figure 19: Hierarchy of UI Elements for Customization of Genesys Supervisor Desktop

Figure 19 shows the object hierarchy of UI elements that are available for customization in GSD. The numbers in Figure 19 indicate the UI elements as follows:

1. This area can be customized by using the View Profile Templates and Object Sets that are enabled through the View Profile Template.
2. This area defines the list view of the View Set.
3. This area defines the detail view of each item in the section. It is the result of applying view templates from the first and second parts of customization.
4. Each view template is displayed in a tab that contains a list view.

View Profile

View Profile Templates are sets of object sets for different resource types. There can be several View Profile Templates plus the default View Profile Template. Object sets that are enabled in View Profile Templates define what is displayed in area 1 of Figure 19 and Figure 20 on page 138.

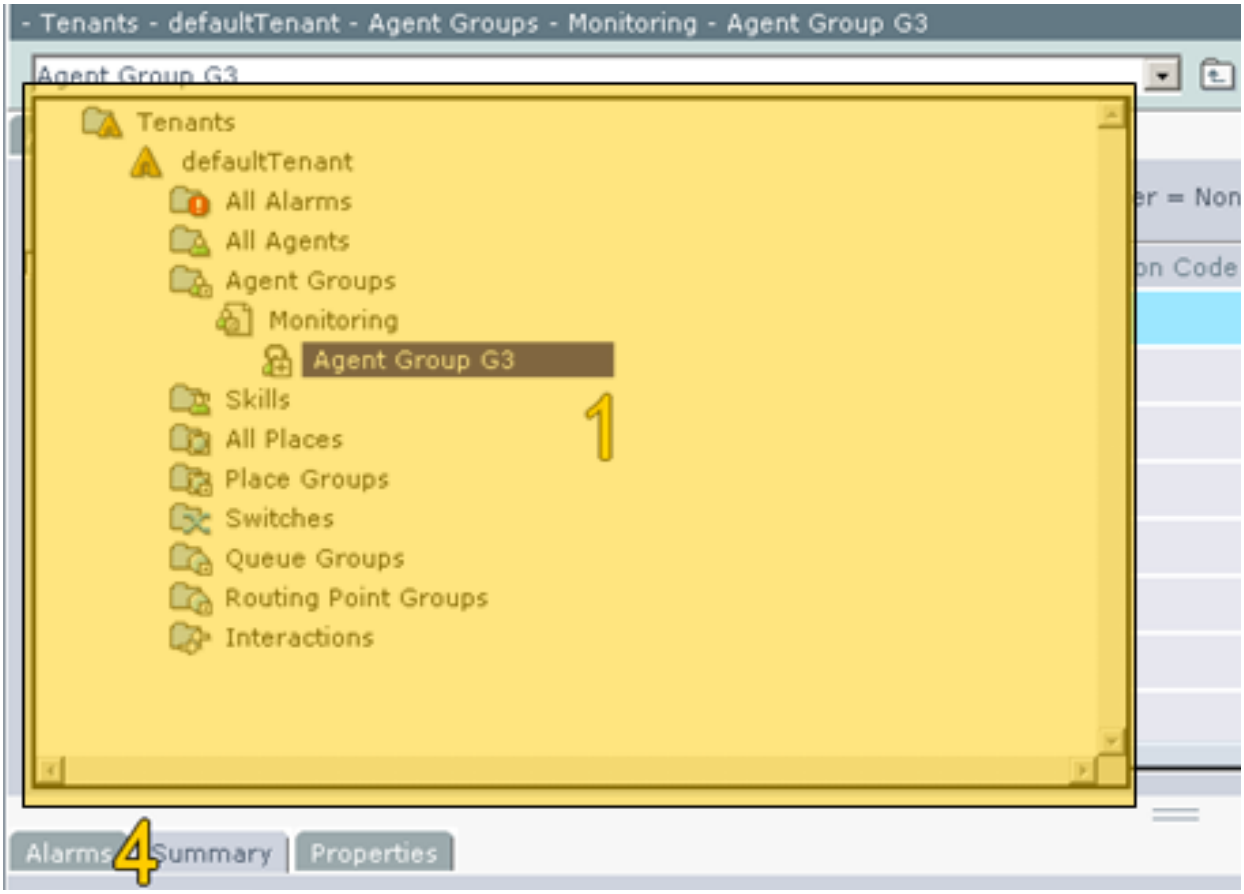


Figure 20: Object sets that are enabled in View Profile Templates define what is displayed in area 1 of the GSD GUI.

Default View Profile

The default View Profile Template *must* always exist. Ensure that you can restore it in the event that it is lost or corrupted. The default View Profile Template is the following file:

ViewProfiles.xml `../supervisor/data/ViewProfiles.xml`

`ViewProfiles.xml` can be set or modified by using the View Profile Wizard. The default file is shown here:

```
<viewProfileDefinitions>
   <viewProfile
      name="Default"
      id="Default"
      description="Default View Profile" >
      <objectSets>
         <objectSet id="All Tenants" />
         <objectSet id="All Interaction Queues" />
      </objectSets>
```

```
   </viewProfile>
</viewProfileDefinitions>
```

View Profile Template Wizard

The View Profile Template Wizard is accessed from the View Profile Templates window. From the GSD GUI, select `Supervisor>Settings>View Profile Templates` from the action bar (see *Genesys Supervisor Desktop Help*). Figure 21 shows the first panel of the View Profile Template Wizard.

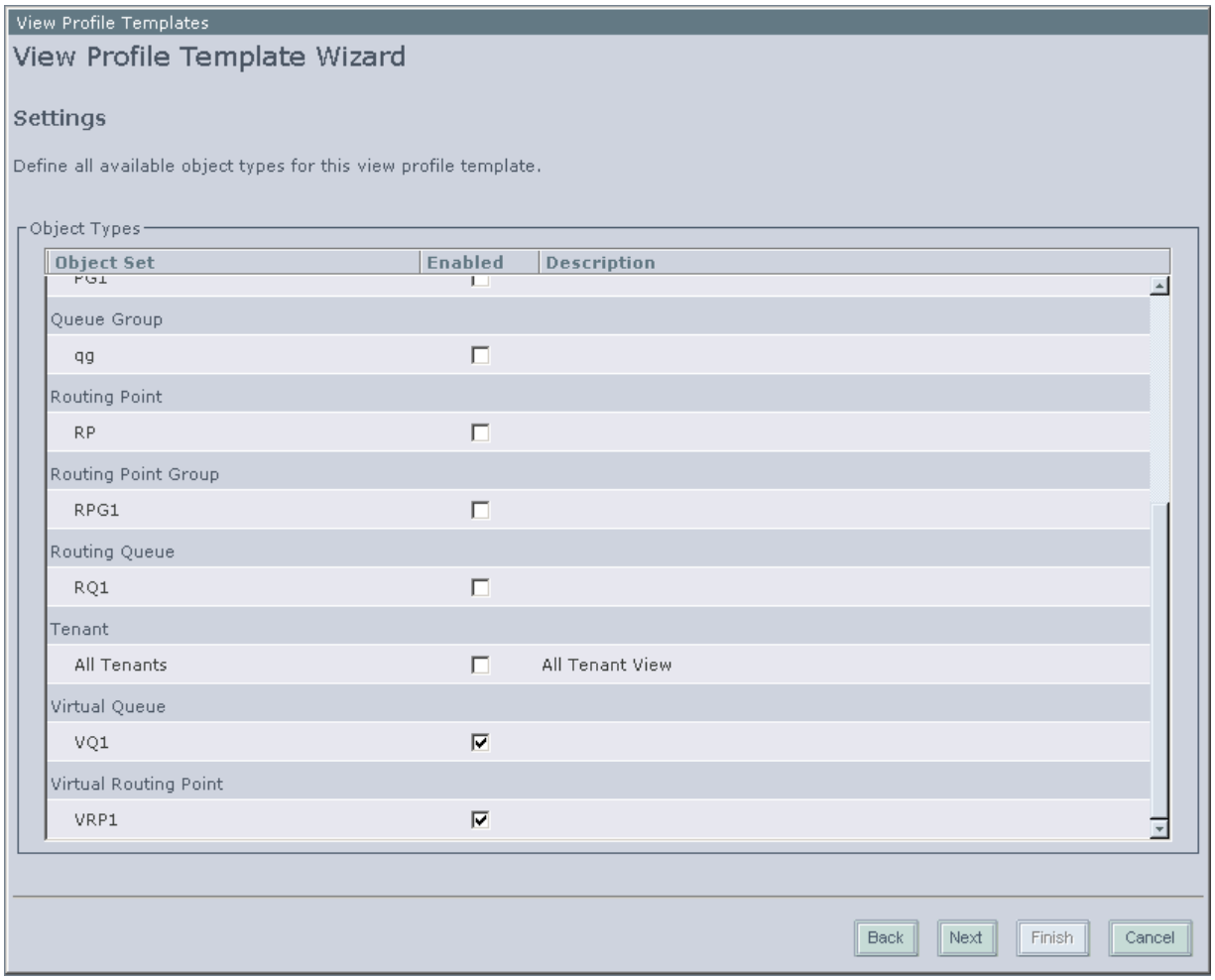


Figure 21: The first panel of the View Profile Template Wizard. This wizard is used to configure the View Profile Template, a set of object sets for different resource types and custom hierarchy groups.

Step through the View Profile Template Wizard to specify settings for object set hierarchy groups (see Figure 22 on page 140 and *Genesys Supervisor Desktop Help*).

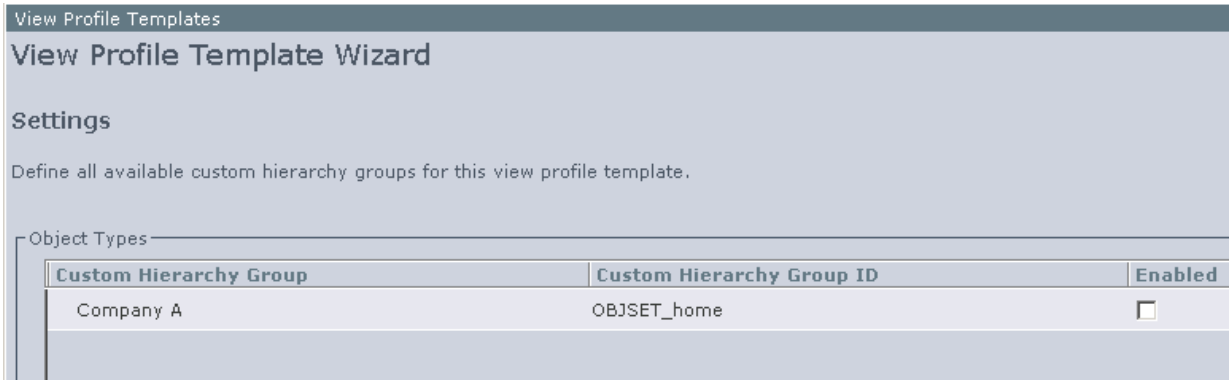


Figure 22: The View Profile Template Settings panel. This panel specifies a set of object sets for different resource types and custom hierarchy groups.

Object Set

Figure 23 shows the Object Sets for the various resource types in area 1 of the GSD GUI. Object Sets include Alarms, Groups, Places, and so on.

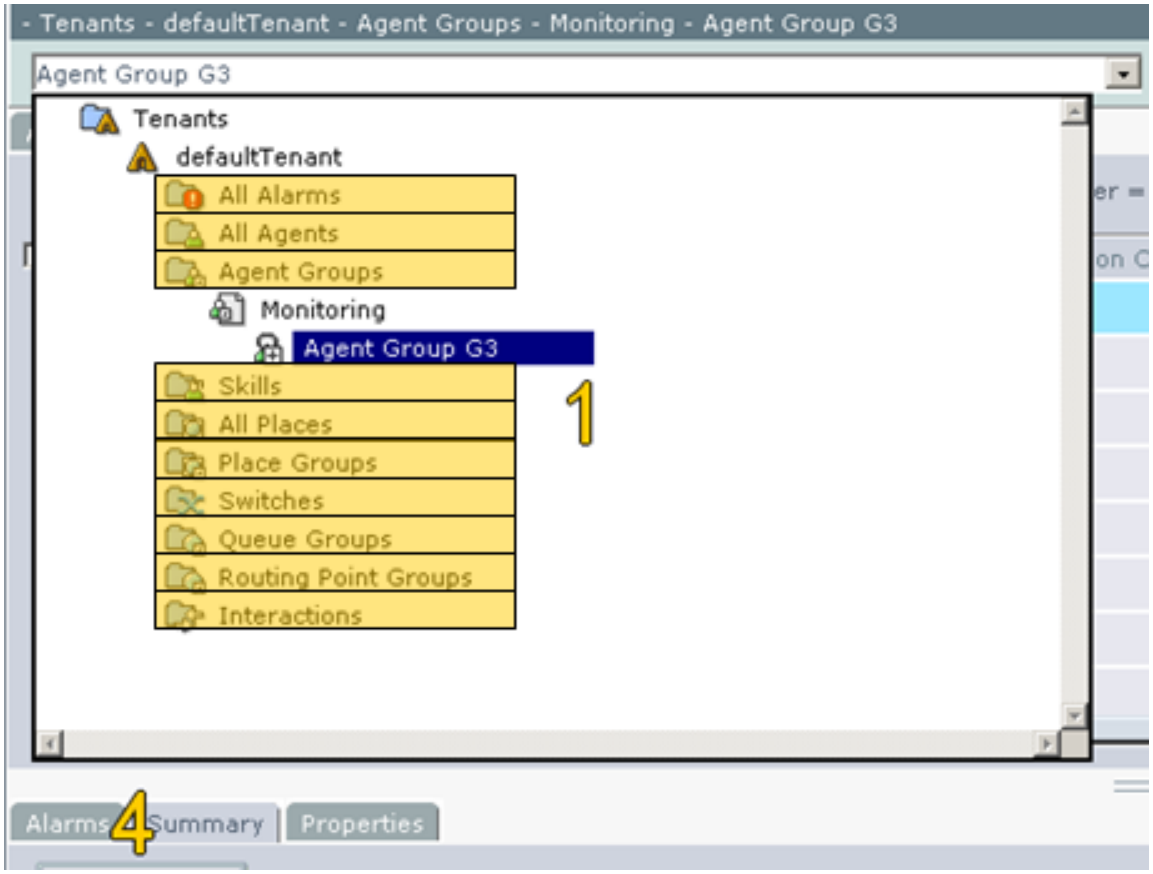


Figure 23: An Object Set is a set of objects of a given resource type.

Default Object Set

The default View Profile Template contains two object sets: `All Tenants` and `All Interaction Queues`.

The `ObjectSets.xml` file is found in the `supervisor/data/` directory. This file can be configured through the Object Set Wizard. The following is the default `ObjectSets.xml` file:

```
<objectSetDefinitions>
   <objectSet  id="All Tenants"
        name="All Tenants"
        description="All Tenant View"
        objectType="Tenant"
        groupView=""
        objectView="All Tenants"
   />

   <objectSet  id="All Interaction Queues"
        name="All Interaction Queues"
        description="All Interaction Queues View"
        objectType="InteractionQueue"
        groupView="gl_All Interaction Queues"
        objectView="ol_All Interaction Queues"
   />

</objectSetDefinitions>
```

Object Set Wizard

Step through the Object Set Wizard to specify the set of objects for each Object Set, and to specify the type of view (List View and/or Summary View) for the Group Level and for the Object Level (see Figures 24, 25, and 26, as well as *Genesys Supervisor Desktop Help*).

Objects View Template

The Object Set Wizard can be used to specify the collection of objects that is displayed when the Supervisor selects the Object Set. From the GSD GUI, select `Supervisor>Settings>Object Sets` from the action bar (see *Genesys Supervisor Desktop Help*). Figure 24 on page 142 shows the first panel of the Object Set Wizard.

Figure 24: First panel of the Object Sets Wizard. An Object Set is a set of objects of a given resource type.

Group Level View Templates

The Object Set Wizard can be used to specify the type of view for each View Template. The Supervisor Work Area provides two types of views, List Views and Summary Views. In a List View, one View Template is used to display an array of statistics and properties for a list of objects. In the Summary View, a collection of View Templates is used to display a complete listing of statistics and properties for a single object (see *Genesys Supervisor Desktop Help*).

Figure 25 on page 143 shows the View Templates - Group Level panel of the Object Set Wizard. Select the views that are to be available when the end user displays the object groups that reside in this Object Set. For example, Agent Groups and Place Groups are group objects. If necessary, modify the list of View Templates.

Figure 25: Group level Object Set Wizard. The Object Set wizard can define what list (view) type is visible for both group level and object level.

Object Level View Templates

The Object Set Wizard can be used to specify the type of view for each Object. The Supervisor Work Area provides two types of views: List Views and Summary Views. In a List View, one View Template is used to display an array of statistics and properties for a list of objects. In a Summary View, a collection of View Templates is used to display a complete listing of statistics and properties for a single object (see *Genesys Supervisor Desktop Help*).

Figure 26 on page 144 shows the View Templates - Object Level panel of the Object Set Wizard. Select the views that are to be available when the end user displays the lower-level objects that reside in this Object Set. For example, Agents and Places are lower-level objects. If necessary, modify the list of View Templates.

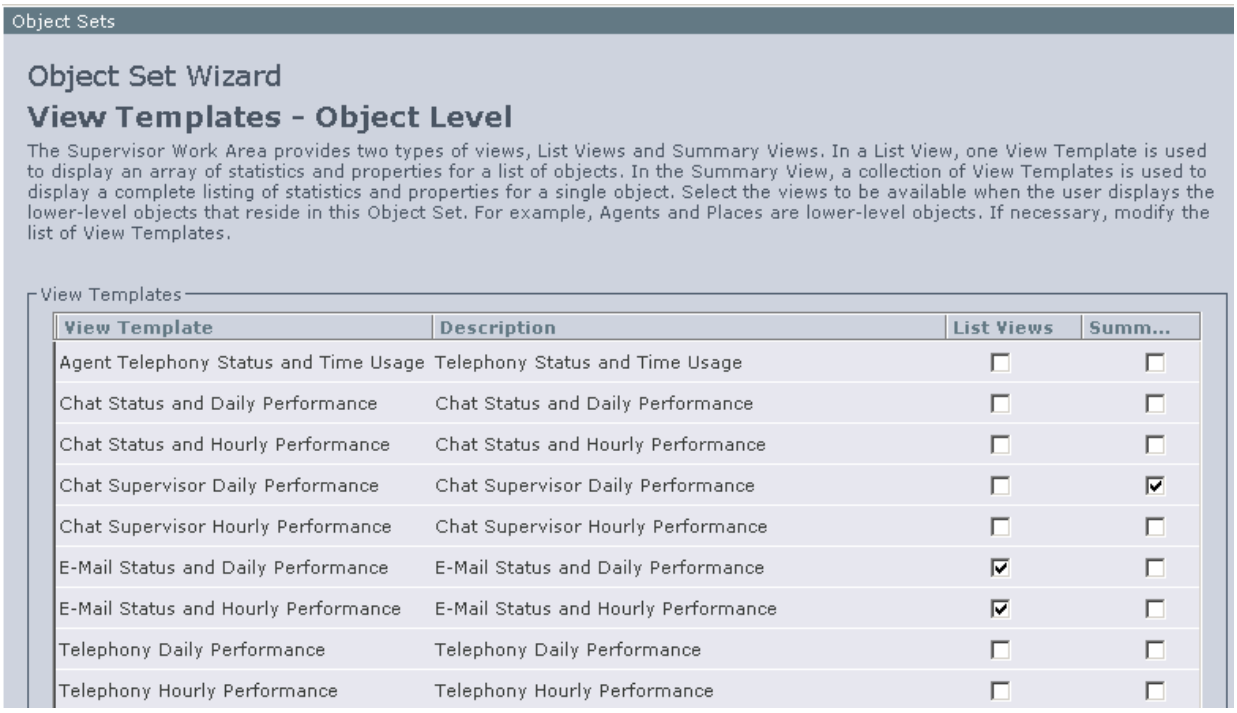


Figure 26: Object level Object Set Wizard. The Object Set Wizard can define what is visible for both group level and object level.

Object Set Definitions: GSDResource.properties File

The GSD Object Sets are defined by the `GSDResources.properties` file, which is found in the following directory:

```
gdesktop\WEB-INF\classes\com\genesyslab\uadthin\supervisor\resources\GSDResources.properties
```

The following is an example of a configured view. The numbers that is associated with each `resource` in each line is the resource type id; this is an internal GSD enumeration. For each resource type you can define different titles; these will be used in various parts of the GUI, the XML file that customizes each view, and images (for an object of that type, and for a list or a folder of objects of that type) that are used in the GUI.

```
resource.list.title.1=Agent List
resource.single.title.1=Agent
resource.title.1=Agents
resource.XML.1=/supervisor/xml/agents.xml
resource.image.1=supervisor/img/ObjectIcons/agent.gif
resource.list.image.1=supervisor/img/FoldersNLists/AgentList.gif
resource.folder.image.1=supervisor/img/FoldersNLists/AgentFolder.gif

...
```

```
resource.list.title.4=Agent Group List
resource.single.title.4=Agent Group
resource.title.4=Agent Groups
resource.XML.4=/supervisor/xml/agtgrps.xml
resource.image.4=supervisor/img/ObjectIcons/agentGroup.gif
resource.list.image.4=supervisor/img/FoldersNLists/AgentGroupList.gif
resource.folder.image.4=supervisor/img/FoldersNLists/AgentGroupFolder.gif

...
```

Object Set Definitions: supervisor/xml/ObjectSets.xml

The hierarchy of views is defined in the `ObjectSets.xml` file. The hierarchy is shown in the Objects drop-down list (see Figure 27 and also the section "Displaying Object Lists and Objects" in *Genesys Supervisor Desktop Help*).

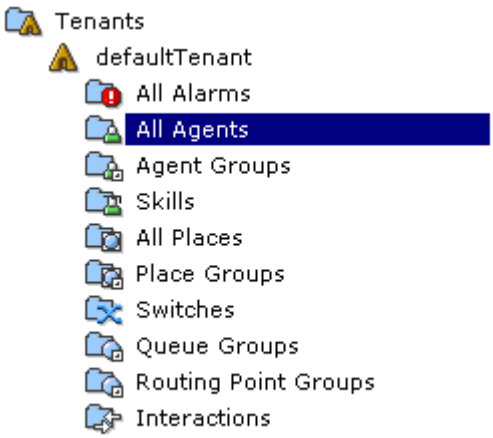


Figure 27: The ObjectSet.xml file controls the resource types that are displayed in this drop-down list.

Note: The content of the drop-down is defined by the View Profile Template and object sets; these are valid for the current supervisor; however, the file `ObjectSet.xml` always has more resource types available than can be seen in this drop-down list.

The `ObjectSets.xml` file defines view sets, titles, and images for default resource types; if these are turned on in the View Profile Template, they are visible in the GSD GUI. The file is found in the following directory:

```
gdesktop\supervisor\xml\objectsets.xml
```

The following is an example of a configured `ObjectSets.xml` file. In this example, the following resource types are used:

- 991—persons
- 1—agents
- 992—all places
- 5—places

Table 7 contains a list of attributes that are used for this example. Each Object Set can have these attributes.

```
<objectset id="root" resource_type='0' title=""
   viewset="/supervisor/xml/otypes.xml" bundle='none' >
<objectset id="tenants" resource_type='11' title="tenantlist.title"
   viewset="/supervisor/xml/tenants.xml" image="img/FoldersNLists/TenantFolder.gif" >

   <objectset id='atenant' resource_type='994' title='resource.title.994'
      viewset="/supervisor/xml/otypes.xml" image="img/ObjectIcons/tenant.gif">

      <objectset id="alarms" resource_type="16" title="allalarms.title"
         viewset="/supervisor/xml/alarms.xml" image="img/FoldersNLists/AlarmsFolder.gif" />

      ...

      <objectset id='e_inter' resource_type="990" title="resource.title.990"
         viewset="/supervisor/xml/etypes.xml" image="img/FoldersNLists/InteractionFolder.gif">
      <objectset id="email_seg" resource_type="996" title="resource.title.996"
         viewset="/supervisor/xml/email_segs.xml"
         image="img/FoldersNLists/EmailCustomerSegmentFolder.gif"/>
      <objectset id="email_que" resource_type="13" title="resource.title.13"
         viewset="/supervisor/xml/email_ques.xml"
         image="img/FoldersNLists/InteractionQueueFolder.gif"/>
      <?includeIF
         condition='com.genesyslab.uadthin.supervisor.view.XMLPIcomponentGTEversion'
         component='IsServer' version='7.1'
         <objectset id="work_bin" resource_type="32" title="resource.title.32"
            viewset="/supervisor/xml/work_bins.xml"
            image="img/FoldersNLists/WorkbinFolder.gif" />
      ?>
      <objectset id="email_svc" resource_type="997" title="resource.title.997"
         viewset="/supervisor/xml/email_svcs.xml"
         image="img/FoldersNLists/EmailServiceTypeFolder.gif"/>
      <objectset id="emails" resource_type="12" title="resource.title.12"
         viewset="/supervisor/xml/emails.xml" image="img/FoldersNLists/InteractionFolder.gif"/>
   </objectset>
</objectset>
</objectset>
```

Table 7: Attributes for the ObjectSets.xml File

Attribute	Description
id	Identifier for this object.
title	Display title. If the bundle is supplied, `title` is used as a key to fetch the text.
bundle	Optional property resource file for titles. It is instantiated only once. The locale is determined by the reader of this file.
image	May be ignored by the JSP file that is building the visuals. If a bundle is specified, `image` may be a key to the bundle that specifies the localized image.
viewset	The file that describes the view set. The reader reads each of these files to describe the object view at that level.
resource_type	Values in the 900 range are place holders and are not associated with real resources.

Object Sets can be nested and they can be wrapped by `includeIF` directives such as in this example:

```
<?includeIF condition='conditionClassName' ...attributes used by condition class...
  <xml fragment used when codition is true>
 %else%
  <xml fragment used when condition is false>
?>
```

The `includeIF` directive is evaluated in the `condition` class, which might use optional attributes and include additional XML code. You can create your custom node with a custom link, if needed

The following is an example of a custom object set. Figure 28 on page 148 shows the results of implementing this file in the following location:

`gdesktop/supervisor/data/ObjectSets.xml`

```
<objectset id="home" resource_type='1000' title="Company A"
   viewset="/supervisor/xml/otypes.xml" bundle='none' >
   <objectset id='custom' resource_type='1001' title='Custom Link'  web_page="http://www.google.com"
      viewset="/supervisor/xml/custom.xml" bundle='none' />
</objectset>
```

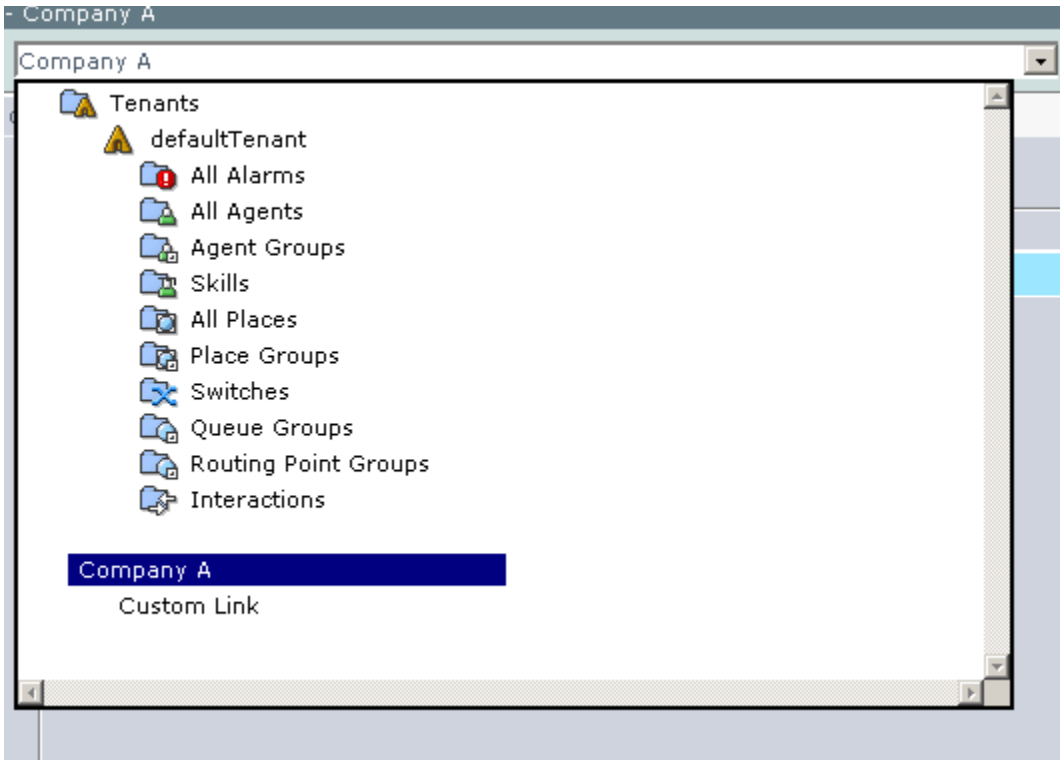


Figure 28: A Custom Object Set Inserted into the View Template Using a Customized Object Set File Appears in the Drop-down List.

Object Views

Object Views are defined by `ViewSets.xml` files. The default object sets are: `all tenants` and `all interaction queues`. Identifiers for `groupView` and `objectView` are also defined in the `ViewSets.xml` file.

Viewsets

Each XML file has the root element `viewset`, which describes the characteristics of a view. In the GSD GUI, the `viewset` comprises the list view (see area 2 in Figure 19) and the detail view (see area 3 in Figure 19). The following is an example of a `viewset` file:

```
<viewset id="agents" title="agents.title"
bundle="com.genesyslab.uadthin.supervisor.resources.GSDResources" >
   <preinclude page="/supervisor/jsp/persons_pre.jsp" />
   <postinclude page="/supervisor/jsp/persons.jsp" />
    <section id="main" selected="agents" listed="alarms, agents">

   ...

   </section>
```

```
   <section id="detail" selected="summary"
listed="almstab,summary,agtprops">

   ...

    </section>
</viewset>
```

Table 8 lists the attributes of each `viewset`.

Table 8: Viewset Attributes

Attribute	Description
id	Object identifier.
title	Display title. If bundle is supplied, then `title` is used as a key to fetch the text.
bundle	Optional property resource file for titles. It is instantiated only once. The locale is determined by the reader of this file.
image	May be ignored by the JSP file that is building the visuals. If a bundle is specified, `image` may be a key to the bundle that specifies the localized image.
hint	Defines the tooltip.
accesskey	Used as a keyboard shortcut to direct focus to the specified key, typically using ALT+<`accesskey`>. In the XML file, if `accesskey` is a single character, it will be used directly; otherwise, it will be used as a lookup into the localization property file to find the single character.

The `viewset` file may include, preinclude, and postinclude, which enables you to specify a header page and footer page for a view. The `viewset` file may have no more than two section elements. If you specify more than two section elements, the third and subsequent sections are ignored.

The `ViewSets.xml` file cannot be set or configured by a wizard; it is an internal layer that resides between object sets and view templates. It is stored in the following directory:

```
supervisor/data
```

The following is an example `viewset` file:

```
<viewsetdefinitions>

   <viewSet id="All Tenants" objectType="Tenant">
```

```
      <listViews>
        <viewTemplate id="Tenant_E-Mail Handling Status" />
        <viewTemplate id="Tenant_E-Mail Handling Hourly Performance" />
        <viewTemplate id="Tenant_E-Mail Handling Daily Performance" />
      </listViews>
      <summaryView>
        <viewTemplate id="Tenant_E-Mail Handling Status" />
        <viewTemplate id="Tenant_E-Mail Handling Hourly Performance" />
        <viewTemplate id="Tenant_E-Mail Handling Daily Performance" />
      </summaryView>
    </viewSet>

    <viewSet id="gl_All Interaction Queues" objectType="InteractionQueue">
      <listViews>
        <viewTemplate  id="interaction queue status" />
        <viewTemplate  id="interaction queue hourly" />
        <viewTemplate  id="interaction queue daily" />
      </listViews>
      <summaryView>
        <viewTemplate  id"interaction queue status" />
        <viewTemplate  id="interaction queue hourly" />
        <viewTemplate  id="interaction queue daily" />
      </summaryView>
    </viewSet>

    <viewSet id="ol_All Interaction Queues" objectType="Interaction">
      <listViews>
      </listViews>
      <summaryView>
      </summaryView>
    </viewSet>

</viewsetdefinitions>
```

Default View Set The `ViewSets.xml` file defines the view sets that are referenced by default object sets. as shown in the following example:

`gdesktop/supervisor/data/ViewSets.xml`

```
<viewsetdefinitions>

    <viewSet id="All Tenants" objectType="Tenant">
      <listViews>
        <viewTemplate id="Tenant_E-Mail Handling Status" />
        <viewTemplate id="Tenant_E-Mail Handling Hourly Performance" />
        <viewTemplate id="Tenant_E-Mail Handling Daily Performance" />
      </listViews>
      <summaryView>
        <viewTemplate id="Tenant_E-Mail Handling Status" />
        <viewTemplate id="Tenant_E-Mail Handling Hourly Performance" />
        <viewTemplate id="Tenant_E-Mail Handling Daily Performance" />
```

```
      </summaryView>
   </viewSet>

...

</viewsetdefinitions>
```

Viewsets: supervisor/data/xml

View sets may include, preinclude, and postinclude, which enables you to specify a header page and a footer page for a view. The `viewset` file may have no more than two section elements. If you specify more than two section elements, the third and subsequent sections are ignored. The following is an example of this:

gdesktop/supervisor/xml files

```
<viewset id="agents" title="agents.title"
   bundle="com.genesyslab.uadthin.supervisor.resources.GSDResources" >

   <preinclude page="/supervisor/jsp/persons_pre.jsp" />
   <postinclude page="/supervisor/jsp/persons.jsp" />

   <section id="main" selected="agents" listed="alarms, agents">
   ...
   </section>

   <section id="detail" selected="summary" listed="almstab,summary,agtprops">
   ...
   </section>

</viewset>
```

Section

View sets consists of the following two sections: `list` and `summary/detail`. Each section is composed of the `default-menubar` and `viewtemplate` elements. In the GSD GUI, the view set comprises the list view section (see area 2 in Figure 19) and the detail view section (see area 3 in Figure 19). The following is an example of a `section` file:

gdesktop/supervisor/xml files

```
<section id="main" selected="agents" listed="alarms, agents">
   <default-menubar>

   ...

   </default-menubar>
   <viewtemplate id="alarms" handler='alarms' data='unique' title="alarm.title">

   ...
```

```
  </viewtemplate>
  <viewtemplate id="agents" title="agents.title">

  ...

  </viewtemplate>
</section>
```

Table 9 shows the `section` attributes.

Table 9: Section attributes

Attribute	Description
id	Identifier of this object.
selected	Selected `viewtemplate` element from listed.
listed	`viewtemplate` elements that are present in this section.
header	Included as the header in this section.
footer	Included as the footer in this section.
class	Indicates the `OPNSectionInterface` class other than `OPNSectionImpl`, which is used by default.

View template

The `viewtemplate` element may include `menubar`, `dataitem`, `group`, `propertygroup` or `childview` elements.

Default View

The `ViewTemplates.xml` file can be modified and configured by the View Template Wizard. The following is the default `ViewTemplates.xml` file:

`supervisor/data/`

```
<viewtemplatedefinitions>

  <viewtemplate id="Queue_Status and 15 Minutes"
          name="Status and 15 Minutes Performance"
          displayName="Status and 15 Minutes"
          Description="Status and 15 Minutes Performance"
          objecttypes=
 "Queue,QueueGroup,RoutingPoint,RoutingPointGroup,RoutingQueue,VirtualQueue,VirtualRoutePoint">
    <initialProperties>
      <viewdataitem id="c_Name" width="20" />
    </initialProperties>
    <statgroup name="Status">
      <statdataitem id="s_CallsInQueue" width="5" precision="0" />
      <statdataitem id="s_EstWaitTm" width="8" precision="0" />
```

```
        <statdataitem id="s_MaxWaitTm" width="8" precision="0" />
      </statgroup>
      <statgroup name="15 Minutes">
        <statdataitem id="s_SvcFactor(15min)" width="5" precision="0" />
        <statdataitem id="s_Distrib(15min)" width="5" precision="0" />
        <statdataitem id="s_PctDistrib(15min)" width="5" precision="0" />
        <statdataitem id="s_AvgDistribTm(15min)" width="5" precision="0" />
        <statdataitem id="s_Abandoned(15min)" width="5" precision="0" />
        <statdataitem id="s_PctAbandoned(15min)" width="5" precision="0" />
        <statdataitem id="s_AvgAbandonedTm(15min)" width="5" precision="0" />
      </statgroup>
    </viewtemplate>

    ...

</viewtemplatedefinitions>
```

Figure 29 on page 154 displays the view template tabs at areas 1, 2, 3, 4, and 5. Area 6 indicates the list view and area 7 indicates the detail view; these are the view templates. Each section consists of several view templates that are navigable by using tabs. In Figure 30 on page 155, the `Summary` tab is composed of several view templates, each of which corresponds to one tab in the list view:

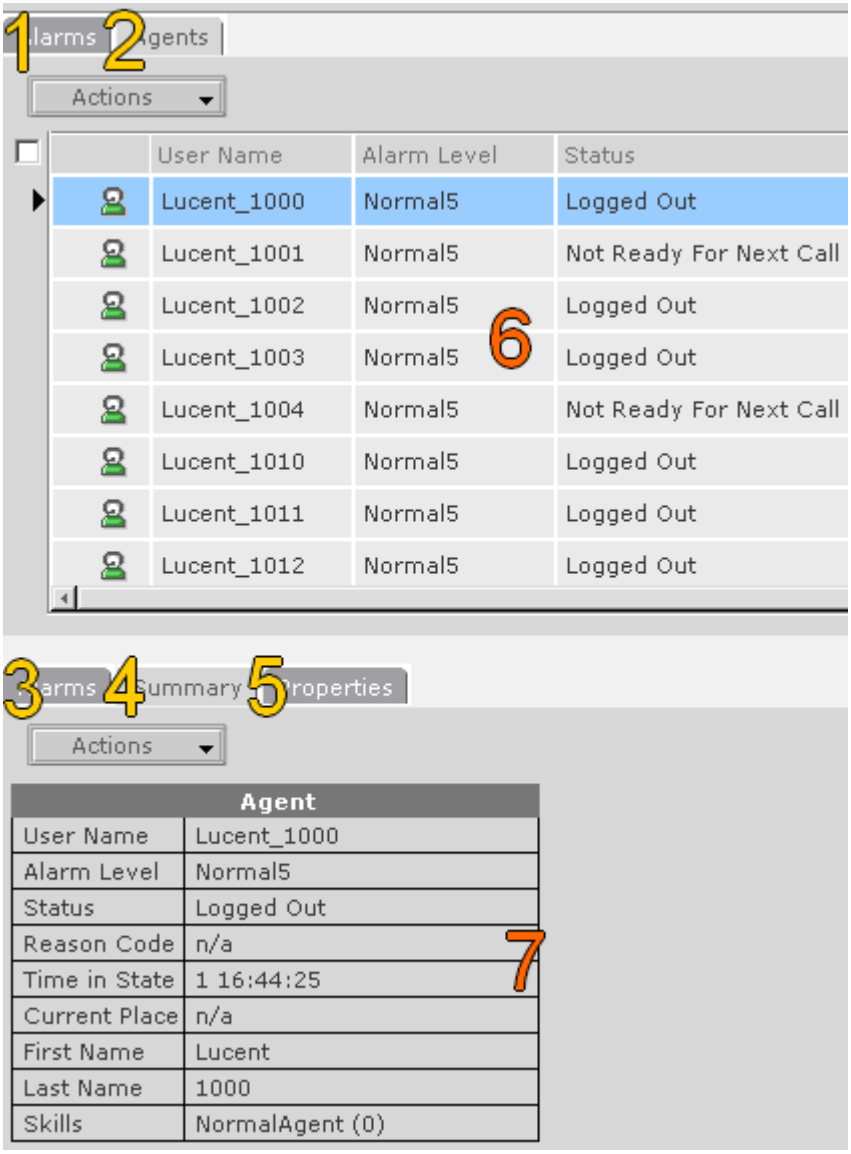


Figure 29: Section Tabs (Areas 1 - 5), List View (Area 6), and Details View (Area 7)

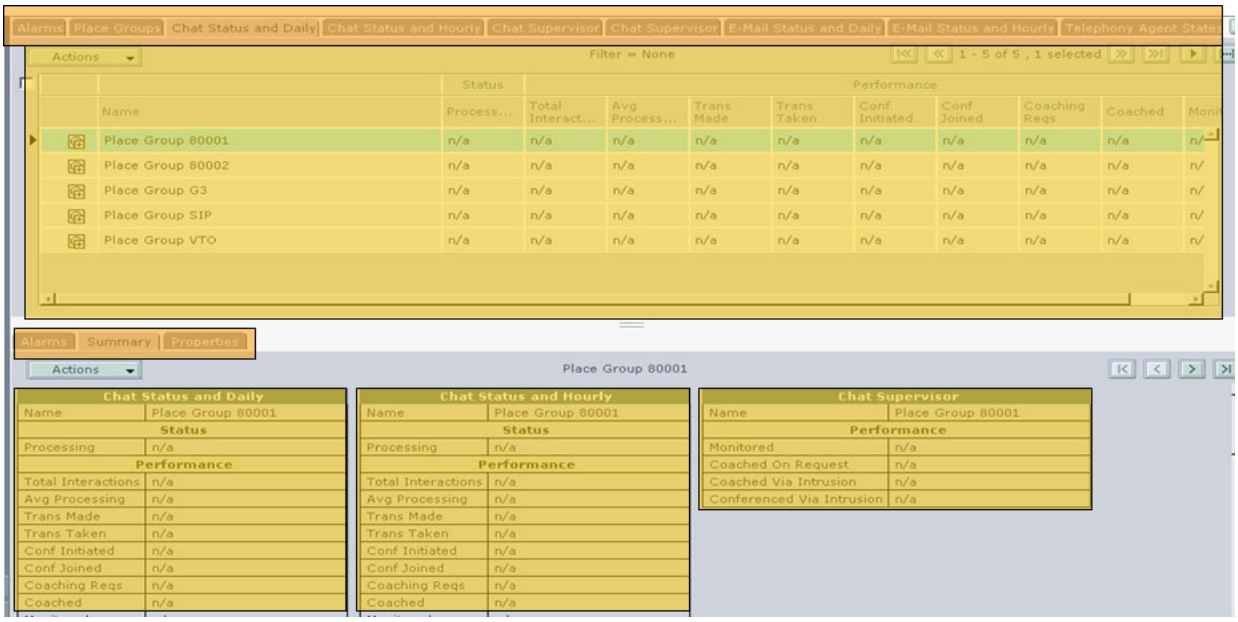


Figure 30: The Summary tab is composed of three view templates in this example. Figure 27 shows a single view template (area 7)

View Template Wizard

A View Template can be defined for several resource types by using the View Template Wizard. A View Template defines an array of statistics and properties that can be displayed in a Supervisor Work Area table. In the first panel of the View Template Wizard, select the appropriate type of contact-center object, then modify the list of View Templates as necessary (see Figure 31 on page 156).

Figure 31: The View Templates Wizard object type selection panel. The Agent Group contact-center object is selected.

The `Name`, `Description`, and `Object Types` panel of the `View Template Wizard` (see Figure 32 on page 157) enable you to enter a unique name, a display name, and a brief description for a new view template. You can then use the check boxes to select the appropriate contact-center object types for the view template.

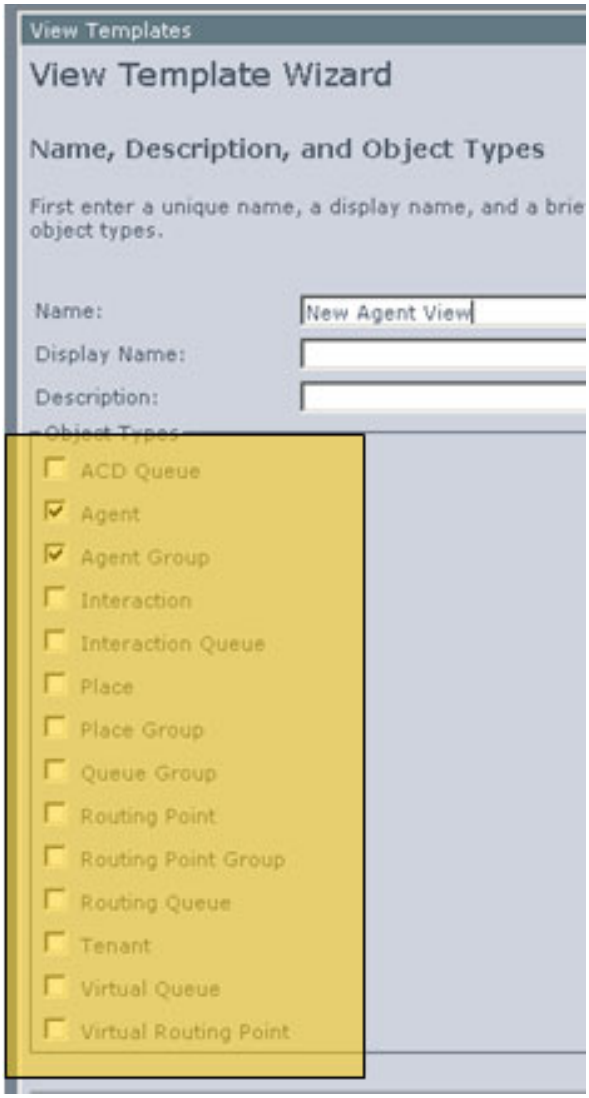


Figure 32: The Name, Description, and Object Types View Template Wizard Panel

In the View Template Wizard, the view template is defined as a set of formattable properties and statistics. Properties are placed at the beginning of the template. By using the tables in Figure 33 on page 158, you can define the properties that are to be included in the template. If necessary, modify the list of available properties by using the controls in the `Selected Properties` table.

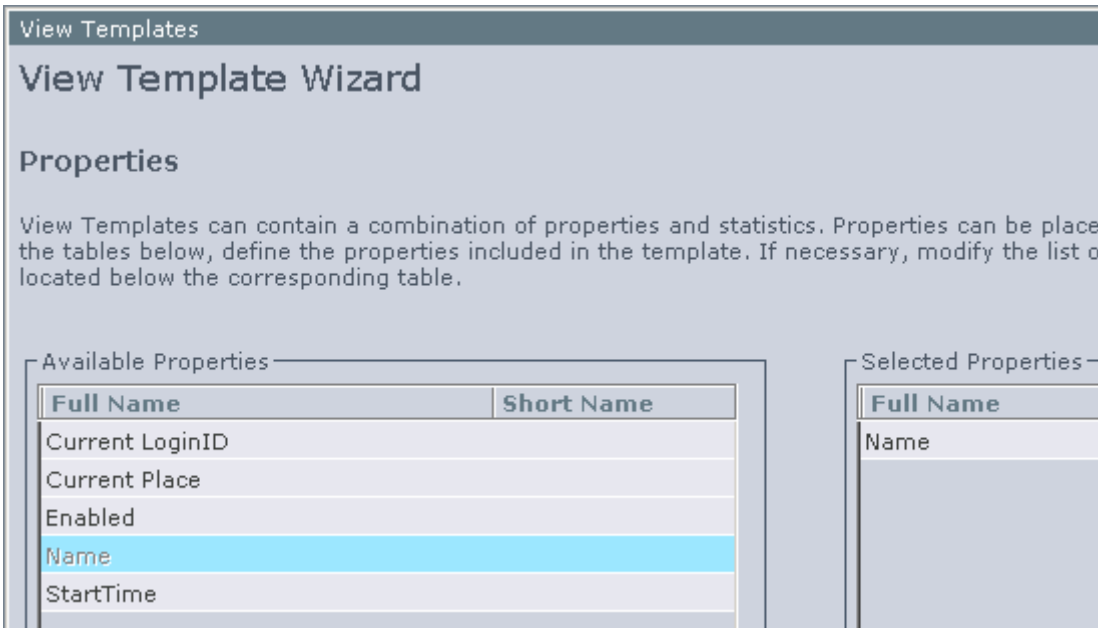


Figure 33: The Properties panel of the View Template Wizard enables you to define the properties of the view template.

Statistics are placed at the end of the template. By using the tables in Figure 34, you can define the statistics that are to be included in the template. If necessary, modify the list of available statistics by using the controls in the `Selected Statistics` table.

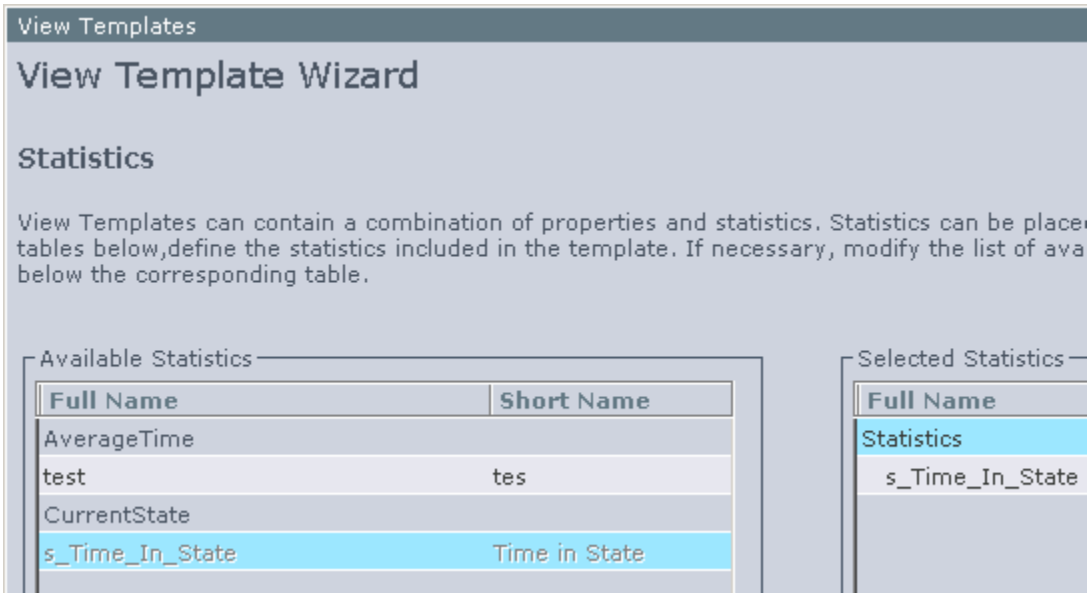


Figure 34: The Statistics panel of the View Template Wizard enables you to define the statistics of the view template.

Use the `Table Column Widths and Level of Precision` panel of the View Template Wizard (see Figure 35) to define the default minimum width of each table column and the level of precision for each entry.

Note: If all columns fit the screen width, they will be distributed proportionately.

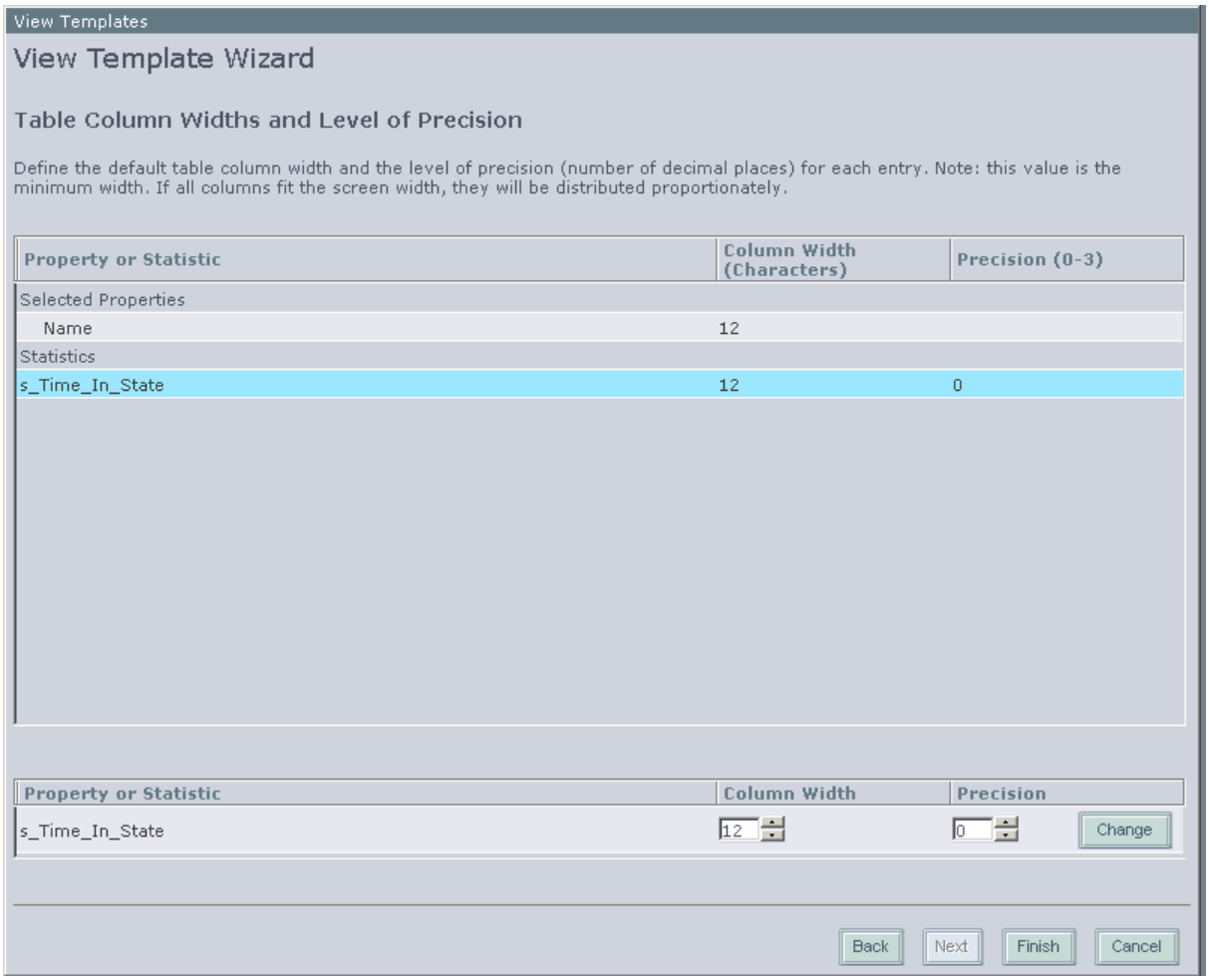


Figure 35: Table Column Widths and Level of Precision Panel of View Template Wizard

View Templates in View Set XML Files

The `ViewTemplates.xml` file defines the view templates that are referenced by the default view sets. as shown in the following example:.

`gdesktop / supervisor / data / ViewTemplates.xml`

```
<viewtemplatedefinitions>

   <viewtemplate id="Queue_Status and 15 Minutes"
      name="Status and 15 Minutes Performance"
      displayName="Status and 15 Minutes"
```

```
      Description="Status and 15 Minutes Performance"
      objecttypes="Queue,QueueGroup,RoutingPoint,RoutingPointGroup">
      <initialProperties>
        <viewdataitem id="c_Name" width="20" />
      </initialProperties>
      <statgroup name="Status">
        <statdataitem id="s_CallsInQueue" width="5" precision="0" />
        ...
      </statgroup>
      <statgroup name="15 Minutes">
        <statdataitem id="s_SvcFactor(15min)" width="5" precision="0"/>...
      </statgroup>
   </viewtemplate>
   ...

</viewtemplatedefinitions>
```

The `viewtemplate` element may include `menubar`, `dataitem`, `group`, `propertygroup`, or `childview` elements, as shown in the following example:

gdesktop/supervisor/xml files

```
<viewtemplate id="alarms" handler='alarms' data='unique' title="alarm.title">
      <menubar>

      ...

      </menubar>
      <dataitem width="20%" align='left' id="a_AllAlarms" title="a_AllAlarms"/>
      <group>

      ...

      </group>
      <propertygroup id="standardSumm" title="agent.title">

      ...

      </propertygroup>
      <childview id="table" default='true'
   class="com.genesyslab.uadthin.supervisor.view.OPNTableTemplateImpl"
   customizer="tableView.jsp">

      ...

      </childview>

   </viewtemplate>
```

"Viewtemplate Attributes" on page 161 contains the attributes of the `viewtemplate` element.

Table 10: Viewtemplate Attributes

Attribute	Description
id	Identifier of the object
data	Either has a unique value or is absent. If it has a unique value, then the data can share the container of other tabs. If it has a non-unique value, the data are obtained from the `viewset` node data so that all templates of the `viewset` share the same filtered container. In the case of a unique value, an additional attribute handler is required to create the container.
handler	Only alarm values can occur. This is reflected in the code of `OPNViewTemplateImpl` which only permits the display of alarms.
class	Indicates the `OPNViewTemplateInterface` class other than `OPNTableTemplateImpl` used by default.
customizer	The page that is used for representing this object.
mediaType	The media type for which the given `viewtemplate` is shown.

UI Elements

The GSD UI elements include: `group`, `property group`, `dataitems`, `childviews`, `dataitem`, `default-menubar`, `menubar`, `menu`, and `action`.

group

The `group` element aggregates other `groups` or `childviews` inside itself. The `group` element is used within the `Property` tab and `Summary` tab to subgroup data visually. The following is an example of a `group` element:

`gdesktop/supervisor/xml files`

```
<group>
  <group>
  ...
  </group>
  <childview>
  ...
  </childview>
</group>
```

property group

The `Agent Properties` tab is used to view and change the configuration properties of a selected Agent (see Figure 36 on page 162). You can customize

the contents of this table to display any or all of the following agent property groups: `General`, `Agent Information`, `Supervisor`, `Skills`, `Login IDs`, and `Agent Groups`.

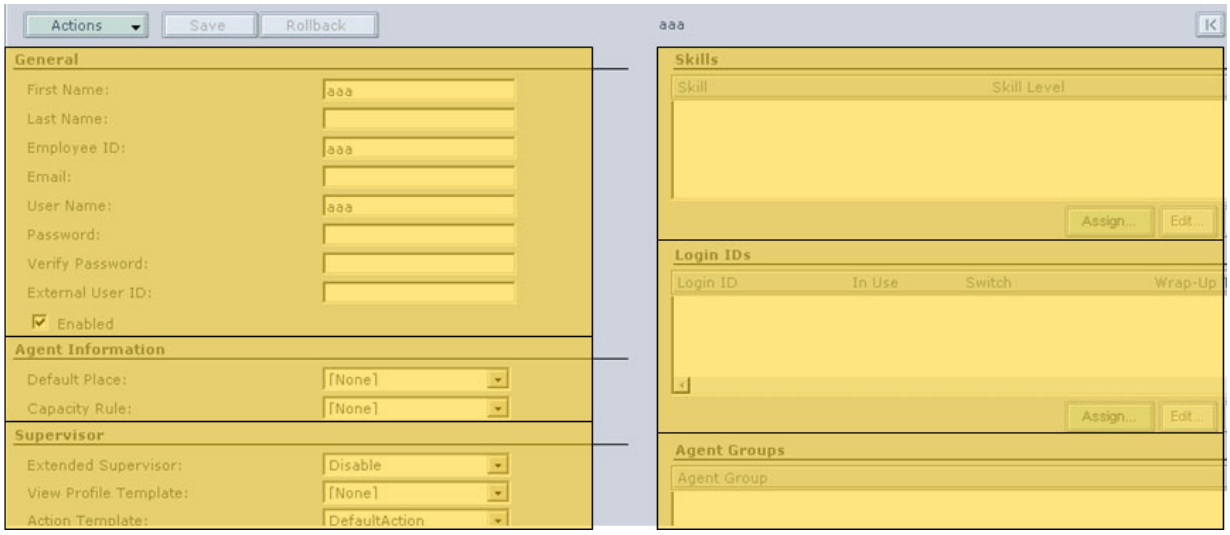


Figure 36: Property groups: General, Agent Information, Supervisor, Skills, Login IDs, Agent Groups

The `Property group` view set XML files are located here:

`gdesktop/supervisor/xml files`

```
<propertygroup id="general" title="general.title">
   <dataitem id="c_FirstName" title="c_FirstName" datatype="string" range="1|32"/>
   <dataitem id="c_LastName" title="c_LastName" datatype="string" range="1|32"/>
   <dataitem id="c_UserName" title="c_UserName" datatype="string" range="1|32"/>
   <dataitem id="c_Password" title="c_Password" datatype="password" range="1|32"
   customizer='passwordProp.jsp'/>
   <dataitem id="c_Enabled" title="c_Enabled" datatype="boolean"/>
</propertygroup>

<propertygroup id="agtinfo" title="agtinfo.title">
   <dataitem id="c_AgentPlace" title="c_AgentPlace" datatype="dropdown"
      customizer="places.jsp"/>
   <dataitem id="c_AgentCapacityRule" title="c_AgentCapacityRule" datatype="dropdown"
      customizer="agtCapRules.jsp"/>
</propertygroup>
```

dataitems The `propertygroup` object includes only `dataitems`. Detail view (7) in Figure 36 is an example of the `propertygroup` object.

```
<propertygroup id="standardSumm" title="agent.title">
          <dataitem id="c_UserName" title="c_UserName"
datatype="string" range="1|32"/>
          <dataitem id="a_CurrentLevel" title="a_CurrentLevel"/>
```

```
          <dataitem id="r_StaticAgtStatus"
title="r_StaticAgtStatus"/>
          <dataitem id="s_Time_In_State"
title="r_StaticAgtStateAge"/>
   ...
</propertygroup>
```

The `property group` view set attributes are listed in Table 11.

Table 11: Property Group View Set Attributes

Attribute	Description
id	Identifier of this object.
title	The display title. If the bundle is supplied in the parent element, `title` is used as a key to fetch the text.
class	Indicates `OPNViewTemplateInterface` class other than `OPNTableTemplateImpl` which is used by default.
customizer	The page that is used for representing this object.

Each `propertygroup` object has an optional customizable attribute (`customizer`) to permit a custom JSP file to build just that piece of the pane, instead of the whole pane. A property group customizer of `none` indicates that the `dataitems` of the group should not be displayed automatically by `doPropertyXXX`. A group customizer of `none_2ndCol` indicates that `doProperty2col` should also embed a JSP file to display this information of the group in the second column.

To render `propertygroups` objects, two JSP files are used: `doProperty2Col.jsp` for `persons` and `places` in several views, and `doProperty.jsp` in all the other cases. Each property item can have the customizer (the JSP file, which constructs the list of values for this property in a combo box), if this property has the data type: `dropdown`; in the other case, the property item is rendered as a simple field (often as the text field) inside the general JSP file, `doProperty.jsp`.

childview A `Childview` is the auxiliary structure for building a table with `dataitems` inside the view template (see Figure 37).

	User Name	Status	First Name	Last Name	Employee ID	Skills
	aaa	Logged Out	aaa		aaa	
	Expert_2001	Logged Out	Expert	2001	2001	
	Expert_2002	Logged Out	Expert	2002	2002	
	Expert_2011	Logged Out	Expert	2011	2011	
	Expert_2012	Logged Out	Expert	2012	2012	
	Lucent_1000	Logged Out	Lucent	1000	1000	Agent (0)
	Lucent_1001	Not Monitored	Lucent	1001	1001	

Figure 37: dataitems inside a view template table

childview view set XML files enable you to configure a table with dataitems inside a view template (see the following example):

gdesktop/supervisor/xml files

```
<childview id="table" default='true'
   class="com.genesyslab.uadthin.supervisor.view.OPNTableTemplateImpl" customizer="tableView.jsp">

   <dataitem width="12%" align='left' id="c_UserName" title="c_UserName"/>
   <dataitem width="12%"  align='left' id="r_StaticAgtStatus" title="r_StaticAgtStatus"/>
   <dataitem width="12%" align='left' id="c_FirstName" title="c_FirstName"/>
   <dataitem width="12%" align='left' id="c_LastName" title="c_LastName"/>
   <dataitem width="12%" align='left' id="c_EmployeeID" title="c_EmployeeID"/>
   <dataitem width="40%" align='left' id="c_AgentSkills" title="c_AgentSkills"/>

</childview>
```

Table 12 shows the childview attributes.

Table 12: childview Attributes

Attribute	Description
id	Identifier of this object.
default	Set the current view as default in the parent element.
class	Indicates OPNViewTemplateInterface class other than OPNTableTemplateImpl used by default.
customizer	Page that is used for representing this object.

childview is composed only of dataitems. The List view that is indicated by area 6 in Figure 29 on page 154 is an example of a childview, the auxiliary structure for building a table with dataitems inside a view template. The following example shows how to construct this table:

```
<childview id="table" default='true'
   class="com.genesyslab.uadthin.supervisor.view.OPNTableTemplateImpl"
   customizer="tableView.jsp">
   <dataitem width="10%" align='left' id="c_UserName" title="c_UserName"/>
   <dataitem width="10%" align='left' id="a_CurrentLevel" title="a_CurrentLevel"/>
   <dataitem width="20%"  align='left' id="r_StaticAgtStatus" title="r_StaticAgtStatus"/>
   <dataitem width="10%"  align='left' id="r_AgentReasonCode" title="r_AgentReasonCode"/>
   <dataitem width="10%"  align='left' id="s_Time_In_State" title="r_StaticAgtStateAge"/>
   <dataitem width="10%"  align='left' id="r_StaticPlaceID" title="r_StaticPlaceID"/>
   <dataitem width="30%" align='left' id="c_AgentSkills" title="c_AgentSkills"/>
</childview>
```

Dataitem

The second code example in the "childview" section shows how data items are defined in XML files, using the `dataitem` XML element. Table 13 shows the attributes of the `dataitem` element.

Table 13: dataitem Attributes

Attribute	Description
id	Identifier of this object.
title	Display title. If the bundle is supplied in the parent element, `title` is used as a key to fetch the text.
align	Alignment (left, right, center) of this `dataitem`.
width	Width of the `dataitem` column.
color	Color of the dataitem.
datatype	Data type (`int`, `string`, `label`, and so on) of the `dataitem` (optional).
customizer	The page that is used to represent this object.

Default-menubar, menubar

The `default-menubar` of the `section` element and the `menubar` of the `viewtemplate` element have similar structures; each can consist of both `menu` elements and `action` elements, as this example shows:

```
<default-menubar>
   <menu id="actions" title="action.title" accesskey="action.key">

   ...

   </menu>
</default-menubar>
```

The `default-menubar` is applied to all viewtemplates unless `menubar` is specified. The `menubar` is used for your custom menus. The `view` menu cannot be specified here; it will be derived from your customization. The `view` menu is created based on views that are specified in each View Template. See the following example:

```
<menubar>
   <menu id="actions" title="action.title">

   ...

   </menu>
```

```
   <action id='displayprt' title='displayprt.title'
    invoke="displayPrintable('processDetailAction.do','$(section)','$(template)','$(action)')"
    actionMapping='alarmsprintable' />
</menubar>
```

menu The `menu` item consists of `actions`, `separators`, and nested `menu` elements. All `actions` are menu items. Figure 38 shows a typical nested menu:

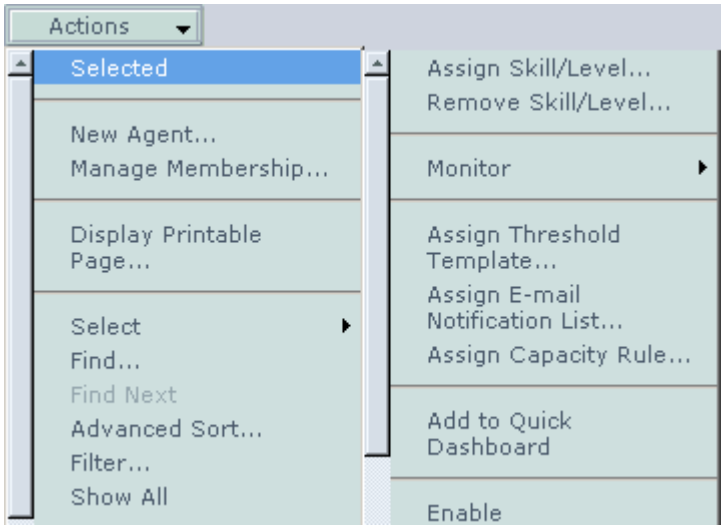


Figure 38: Example GSD GUI menu

The following is an example of `menu` code:

`gdesktop/supervisor/xml files`

```
<menu id="actions" title="action.title" accesskey="action.key">
   <menu id='selected' title='selected.title'>
      <action id='addskill' title='skills.add.title'
         invoke="skillAdd('$(section)','$(template)','$(action)')"
         actionMapping="skillAddRemove"/>
      <action id='removeskill' title='skills.remove.title'
         invoke="skillRemove('$(section)','$(template)','$(action)')"
         actionMapping="skillAddRemove"/>
      <separator />
          <menu id='monitor' title='monitor.title'>
      <?includeIF
   condition='com.genesyslab.uadthin.supervisor.view.XMLPIcomponentGTEversion'
               component='IsServer' version='7.1'
         <action id='monitorchat' title='chat.title' actionMapping="personsAction"/>

         ...

      ?>
      </menu>
```

```
      ...

   </menu>
</menu>
```

menubar Viewset XML Files

The following example shows how the `menu` item is incorporated into the `default-menubar` of the `section` element. The `menubar` of the `viewtemplate` element has a similar structure, being composed of `menu` and `action` elements.

`gdesktop/supervisor/xml files`

```
<menubar>

   <menu id="actions" title="action.title">
   ...
   </menu>
   <action id='displayprt' title='displayprt.title' invoke="displayPrintable(
      'processDetailAction.do','$(section)','$(template)','$(action)')"
      actionMapping='alarmsprintable' />

</menubar>
```

Action

Table 14 summarizes the attributes of the `action` item.

Table 14: action Attributes

Attribute	Description
id	Identifier of this object.
title	The display title. If the bundle is supplied in the parent element, `title` is used as a key to fetch the text.
invoke	A JavaScript function invocation with optional parameters.
actionURI	The attribute value is a URI path, which is used to construct an `ActionForward`.
actionMapping	The attribute value is a struts action mapping, which is used to find a global `ActionForward`.

An `action` can have a CDATA body, which contains a JavaScript function. The invoking of the function is based on the `invoke` attribute; if this attribute is not used, the action is sent to the server, which invokes Java based on the `action` id.

The following example shows the use of the `action` item with a CDATA body:

```
<action id='myact' title='My Title' invoke="doMyAction(event,
'$(action)')">
```

```
   <[CDATA[
      function doMyAction(e, action)
      {
         alert("action "+action+" performed") ;
      }
   ]]>
</action>
```

Table 15 shows the possible substitution values that could be passed inside the JavaScript.

Table 15: Possible Substitution Values that Could be Passed Inside the JavaScript of the Action Item

Value	Description
$(action)	The id of the issued action
$(section)	The section id of the containing section
$(template)	The template id of the containing template
$(form)	The generic form name

Data Items

The following is a list of data items:

- Config items
- Statistics
- Quasi-statistics
- Interaction data items
- Alarm data items
- Third-party data items

Table 16 shows the data type and prefix of each data item as well as where the data item IDs are defined.

Table 16: Data Item Properties

Data Item Types	Prefix	IDs Defined In:
Config items (properties)	c_	Code
Statistics	s_	`userstats.xml`, Code
Quasi-statistics	r_	Code
Interaction data items (properties)	i_	Code

Table 16: Data Item Properties (Continued)

Data Item Types	Prefix	IDs Defined In:
Alarm data items	a_	Code
Third-party data items*	As defined in user data file	User data file

* Third-party data items are described in "Third-Party Data Interface" on page 177.

Config items Config items are the attributes of data items that are inside `property groups` (see Figure 39).

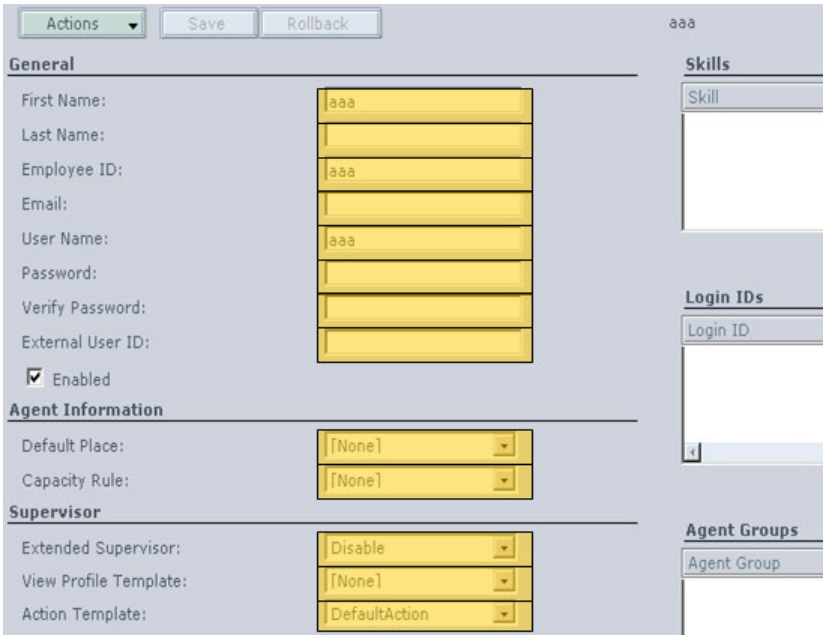


Figure 39: Config Items Inside property groups of GSD GUI

Statistics In the GSD GUI, statistics can be displayed as part of a view template table (see Figure 40).

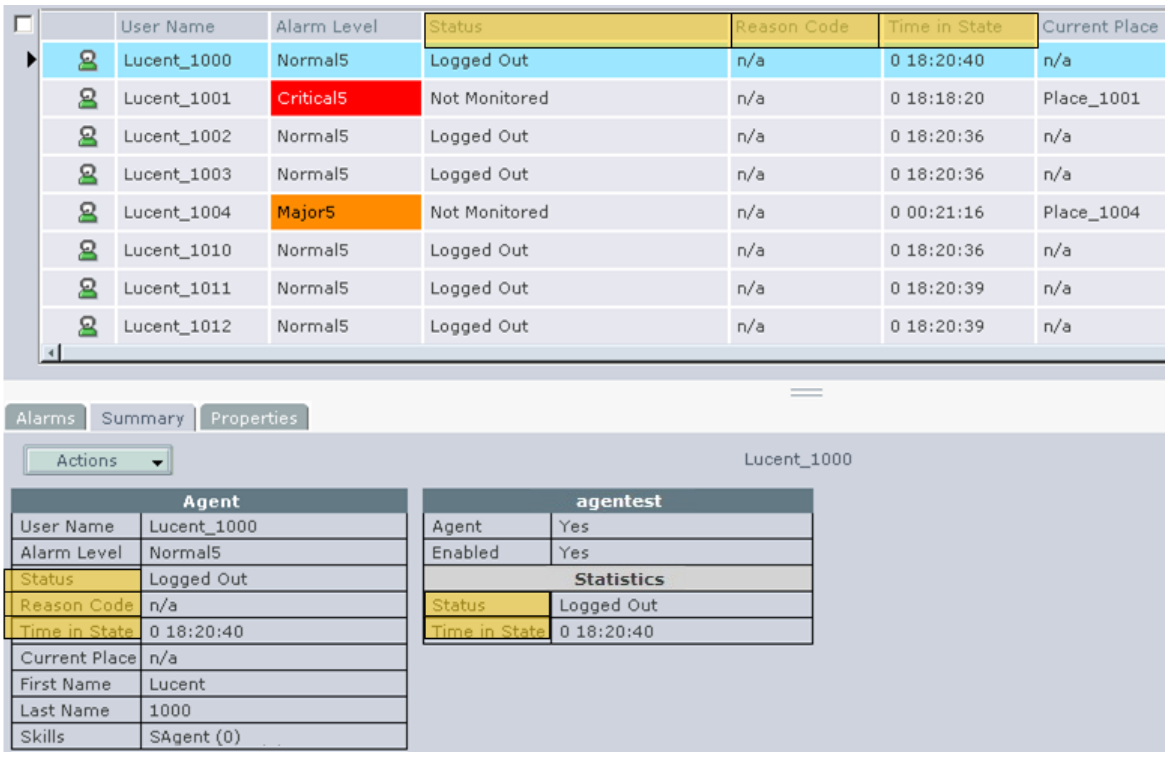


Figure 40: Statistics Within View Template Tables

The `userstats.xml` file can be modified through either the Statistic Wizard or by direct modification of the XML code, as the following example shows:

```
<statgroup id="stats" title="stats.title" >
   <userstat id="s_PctAbandoned"
      fullname="% Abandoned(Daily)"
      alias="% Aband"
      statType="AbandCallsPercentage"
      filter="CALL" timeProfile="daily" />
   <userstat id="s_PctAbandoned(15min)"
      fullname="% Abandoned(15 Min)"
      alias="% Aband"
      statType="AbandCallsPercentage"
      filter="CALL" timeProfile="last15min" />

   ...

</statgroup>
```

In the preceding example, `alias` is used to show a statistic in the GSD GUI. The following attributes are defined by Statistics Server: `statType`, `filter`, and `timeProfile` (see *Statistics Server 7.6 Deployment Guide*).

Statistic Wizard

For each object type in the Supervisor Work Area that has statistics available from Stat Server, you can use the Statistic Wizard to modify the statistics that

can be included in the Supervisor Work Area views (see Figure 41 and *Genesys Supervisor Desktop Help*).

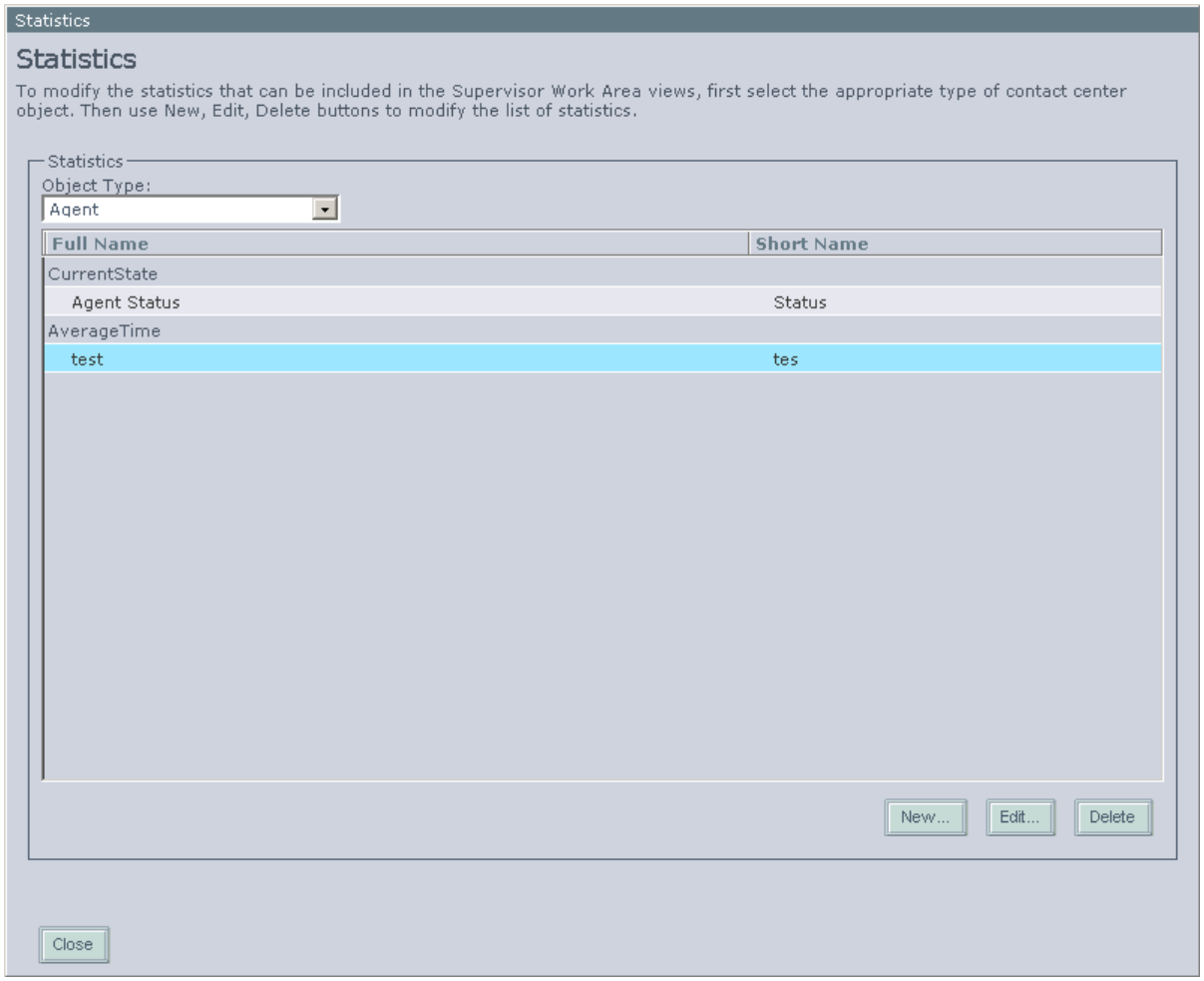


Figure 41: First Panel of Statistic Wizard

In the second panel of the Statistic Wizard, you define the name and settings for a statistic (see Figure 42).

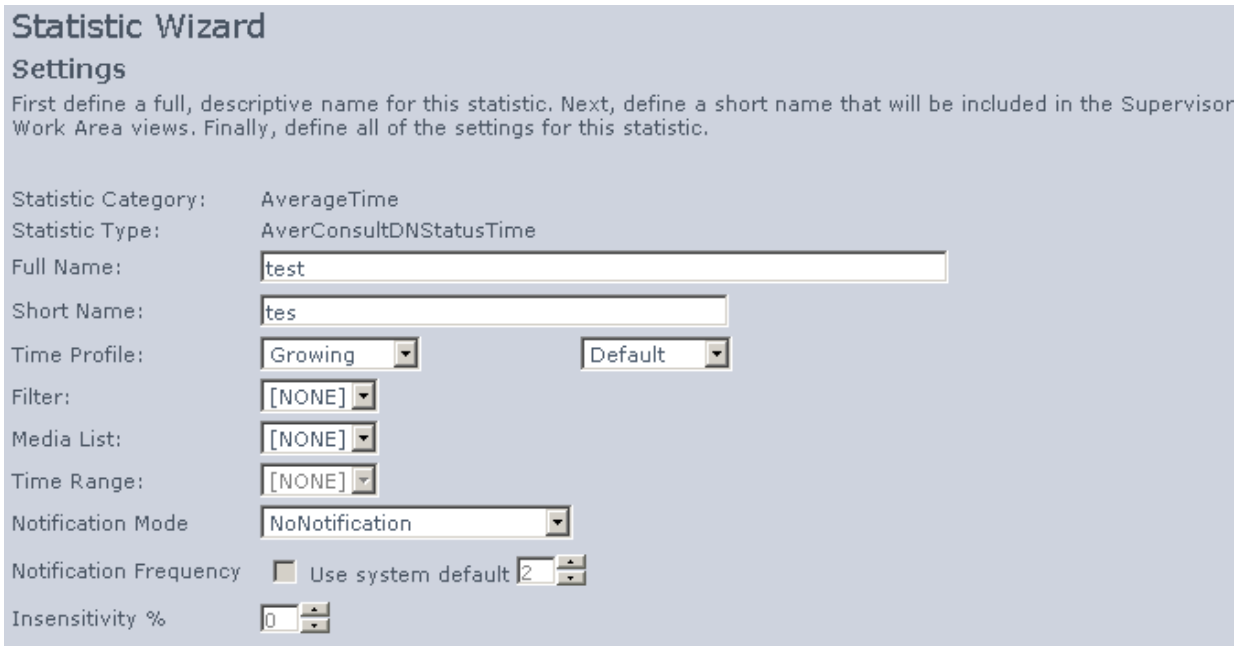


Figure 42: Second Panel of Statistic Wizard

Default Statistics

The `userstats.xlm` XML file defines the default statistics; these are used if no statistics are defined. The alias property is used to show a statistic in the GSD GUI. See the following example:

```
gdesktop/supervisor/data/userstats.xml

<statgroup id="stats" title="stats.title" >

   <userstat id="s_PctAbandoned"
      fullname="% Abandoned(Daily)"
      alias="% Aband"
      statType="AbandCallsPercentage"
      filter="CALL" timeProfile="daily" />
   <userstat id="s_PctAbandoned(15min)"
      fullname="% Abandoned(15 Min)"
      alias="% Aband"
      statType="AbandCallsPercentage"
      filter="CALL" timeProfile="last15min" />
   ...

</statgroup>
```

In the preceding example, `alias` is used to show a statistic in the GSD GUI. The following attributes are defined by Statistics Server: `statType`, `filter`, and `timeProfile` (see *Statistics Server 7.6 Deployment Guide*).

Quasi-Statistics

Quasi-statistics are statistics for which additional calculation is required. For example, `status` could be returned in a low-level form that must be translated

into user-readable form. Like statistics, quasi-statistics are displayed within view template tables (see Figure 43).

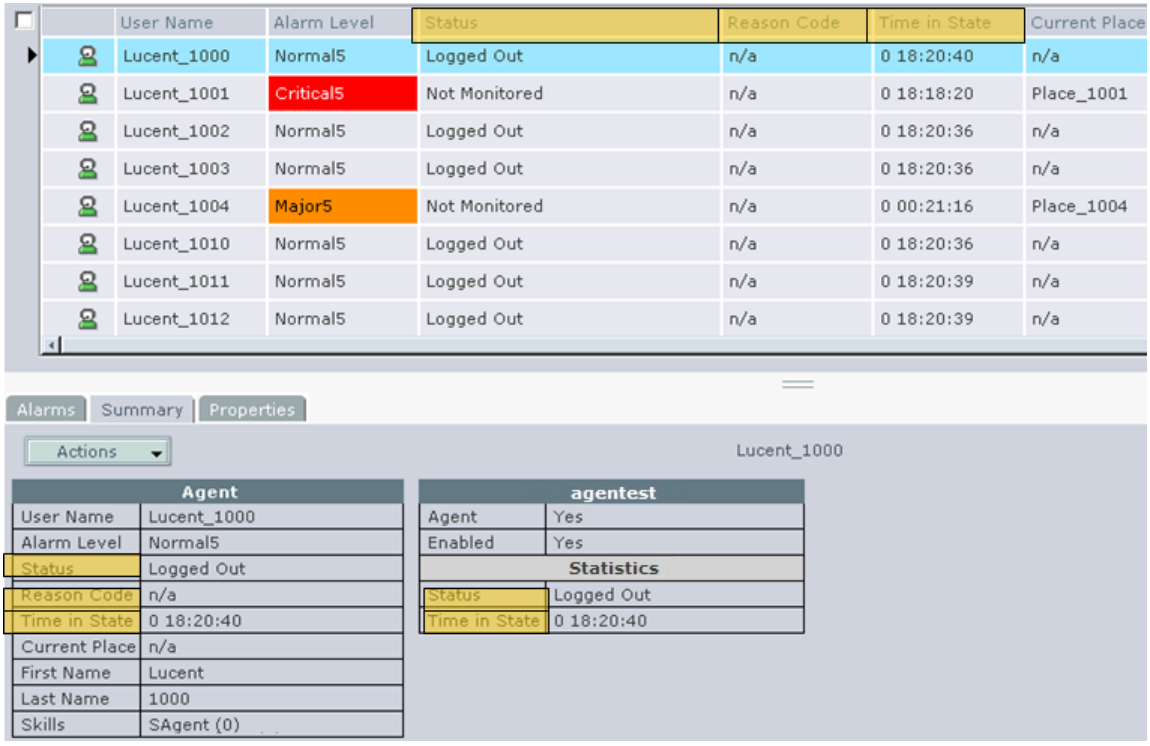


Figure 43: Like statistics, quasi-statistics are displayed within view template tables

Interaction Data Items

Interaction data items are the properties of interactions. These can be seen in the `Property` tab of the Supervisor Work Area (see Figure 44).

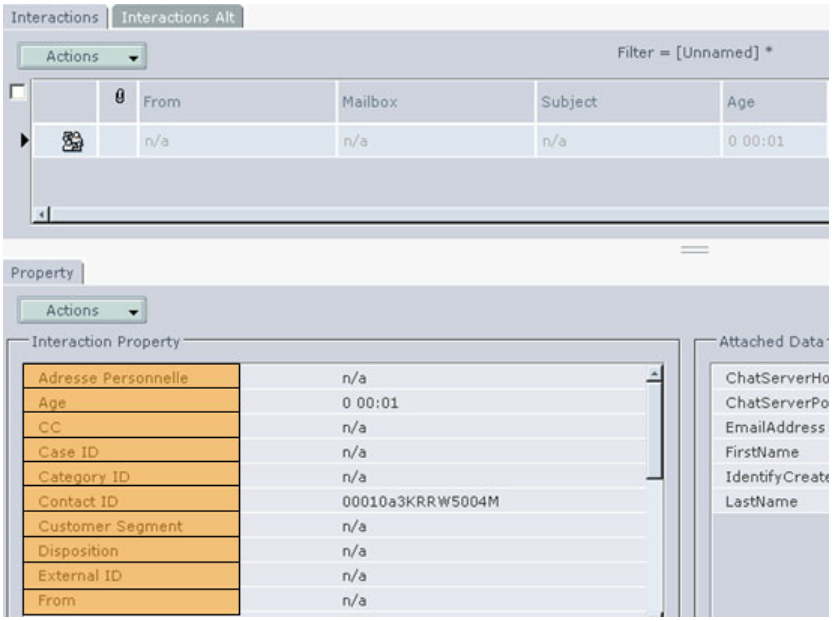


Figure 44: Interaction Property Tab of GSD GUI Showing Interaction Data Items

Alarm Data Items

Alarm data items are special data types that can be displayed in a View Template (see Figure 45 and *Genesys Supervisor Desktop Help*).

Figure 45: Alarm Data Items Displayed Within View Template Tables

Third-Party Data Items

The Third-Party Data Interface (see "Third-Party Data Interface" on page 177) enables you to insert customer-specific data items into GSD views. The data items are updated via HTTP requests. The following is an example of a user data XML file, which is indicated by the name `web.xml`:

```
<dataitems>
  <dataitem>
    <di-fieldID>SalesNA</di-fieldID>
    <di-short-name>Sales NA</di-short-name>
    <di-full-name>Total Sales to North America</di-full-name>
    <di-resource-type>Person</di-resource-type>
    <di-data-type>Float</di-data-type>
    <di-webservice>
      http://ThirdPartyDataServer.abc.com:8080/ThirdPartyDataApp
    </di-webservice>
    <di-keyfield>EmployeeID</di-keyfield>
    <di-listenable>yes</di-listenable>
    <di-sortable>yes</di-sortable>
    <di-filterable>yes</di-filterable>
  </dataitem>
  ...
</dataitems>
```

Migration

Table 17 on page 175 summarizes the things that you should consider when you migrate to the latest Genesys Desktop.

Table 17: Migration Methods and Tasks Related to Customization

Method	Tasks
Wizards	• Wizards store data on ConfigServer, so that this limits issues. • XML files in `gdesktop/supervisor/data` should be copied to the new version.
XML files	• Resources from `gdesktop/WEB-INF/classes/com/genesyslab/uadthin/supervisor/resources/GSDResources.properties` (or localized file) should be copied and pasted into the new version. • XML files in `gdesktop/supervisor/xml` should be copied into the new version. • The XML files in the `data` folder are used for creating the objects in the Configuration Layer only after loading GSD loading onto an empty environment—that is, an empty Configuration Layer scripts folder, or after the Supervisor has used the `Reload example customer view` menu item.
Web services	• Certain lines from the `web.xml` file should be copied and pasted into the new version. • The `userdata.xml` file should be copied into the new version.

IncludeIF Directive

The `includeIF` directive is evaluated in the `condition` class, which can use optional attributes and might or might not include some XML. Table 18 on page 176 lists the attributes of the `includeIF` directive.

`gdesktop/supervisor/xml files`

```
<?includeIF condition='conditionClassName' ...attributes used by condition class...
   <xml fragment used when condition is true>
  %else%
   <xml fragment used when condition is false>
 ?>
```

Table 18: IncludeIF Directive Attributes

Attribute	Description
XMLPIevalCondition	Used by `XMLPIincludeIF` to evaluate an arbitrary condition, to determine whether to include the `true` or `false` part of the PI body
XMLPIcomponentGTEversion	Used to compare the component version for conditional processing instruction
XMLPIareObjectSetsPresent	Checks for the presense of object sets for a specific resource type

Resource Types

Table 19 contains a list of resource types for GSD customization.

Table 19: Customization Resource Types

Number	Name
1	Agent
2	Queue Group
3	ACD Queue
4	Agent Group
5	Place
6	Place Group
7	Template
8	Alarm Definition
10	Switch
11	Tenant
12	Interaction
13	Interaction Queue
14	Enumerator
15	Enumerator Value
16	Alarm

Table 19: Customization Resource Types (Continued)

Number	Name
17	Alarm Aggregate
18	Routing Queue
19	Routing Point Group
20	Virtual Queue
21	Routing Point
22	Virtual Routing Point
23	Skill
26	Login ID
30	Interaction
32	Workbin Type
33	Workbin
990	Interaction
991	Persons
992	Places
993	Switch
994	Tenants
996	Customer Segment
997	Service Type
998	Category

Third-Party Data Interface

The Third-Party Data Interface enables you to insert customer-specific data items into Supervisor Desktop views. The data items are updated via HTTP requests, which are outlined in "Summary of Actions" on page 183.

Note: Within this section, *resources* are equivalent to Genesys configuration objects, and *resource types* are equivalent to Genesys object types.

Defining Third-Party Data to Supervisor Desktop

Note: If you are balancing traffic across multiple Genesys Desktop servers, you must perform these tasks for each instance of the Genesys Desktop application.

Getting Started

1. On your application server system, create the following directory:
   `<INSTALL_PATH>/webapps/gdesktop/custom/`

   For example, if you installed Genesys Desktop on a Windows system, and retained the default installation path, create the following directory:
   `C:\GCTI\GenesysDesktop\webapps\gdesktop\custom\`

   Note: If you have already performed customization tasks for the Agent portion of Genesys Desktop, this directory may already exist.

2. In the `custom/` directory, create a new text file called `userdata.xml`.
3. Stop the Genesys Desktop server if it is running.

   Warning! Do not change the `web.xml` file while the Genesys Desktop server is running.

4. Make these changes to the following Genesys Desktop configuration file:
   `<INSTALL_PATH>/webapps/gdesktop/WEB-INF/web.xml`
   a. In the `<userData>` tag, specify the location of `userdata.xml`. The path must be relative to the `gdesktop/` directory, for example:
   `userData/custom/userdata.xml`
   b. In the `<thirdPartyWebService>` tag, enter the address and port of the web server that supports this instance of the Genesys Desktop application, for example:
   `thirdPartyWebService http://DesktopServer.abc.com:8080`

   For example, if your agents log in to Genesys Desktop using the address `http://DesktopServer.abc.com:8080/gdesktop`, enter `http://DesktopServer.abc.com:8080` in the `<param-value>` tag.

Customizing the Data Item File

For each data item, there is an opening tag and a closing tag. Table 20 on page 179 lists the attributes that are defined within these tags. Three examples of data items are included in "Example XML Data Item Definitions" on page 182.

Note: The XML file is loaded when the Genesys Desktop server is started. Additions, changes, and deletions do not take effect until the Genesys Desktop server is restarted.

Table 20: Data Item File Customization

Tag	Required/Optional	Description
di-fieldID	Required	Enter a unique ID for this data item. This attribute is used to identify the data item, both within the Genesys Desktop web service and within your web service. Do not use any of the following reserved values: • `action` • `listenerID` • `fieldID` • `webService`
di-short-name	Required	Enter a brief name for the data item. This attribute defines the Short Name that is displayed in most Supervisor Work Area views. In general, Short Names should be no more than 16 characters in length.
di-full-name	Required	Enter a full, descriptive name for this data item. This attribute defines the Full Name that is displayed when supervisors hover over a Short Name that is present in a Supervisor Work Area view.

Table 20: Data Item File Customization (Continued)

Tag	Required/Optional	Description
di-resource-type	Required	Specify the type of Genesys resource that is associated with the data item. The valid resource types are listed below: • `Tenant` • `Person` • `AgentGroup` • `Place` • `PlaceGroup` • `Queue` • `VirtualQueue` • `RoutingQueue` • `QueueGroup` • `RoutingPoint` • `VirtualRoutePoint` • `RoutingPointGroup` • `InteractionQueue` **Note:** Resource types are case-sensitive.
di-data-type	Required	Enter the type of data that is returned by your web service. The valid data types are: • `String` • `Integer` • `Float` **Note:** Data types are case-sensitive.
di-webservice	Required	Enter the URL of the web service that will provide the value for this data item, for example: `http://ThirdPartyDataServer.abc.com:8080/ThirdPartyDataApp`. **Note:** Web application names are case-sensitive.

Table 20: Data Item File Customization (Continued)

Tag	Required/Optional	Description
di-keyfield	Required	Enter the field(s) that your web service will use to identify the Genesys resource to which this data item applies. Each keyfield corresponds to a property of the resource. **Keyfields for All Resources:** • `Name` (String) • `Enabled` (String—True or False) **Additional Keyfields for Person Resources:** • `UserName` (String) • `FirstName` (String) • `LastName` (String) • `EmployeeID` (String) • `AgentPlace` (String) **Additional Keyfields for DN Resources:** • `DNNumber` (String) • `LoginFlag` (String—True or False) • `SwitchSpecificType` (Integer) • `SwitchName` (String) **Note:** When you are working with DN resources, the keyfield `Name` returns `Name@SwitchName` in order to ensure uniqueness.
di-listenable	Optional	Enter `yes` or `no`, to specify whether your web service will send an event when the value of the data item changes. The default value is `no`. For further information, see "Actions for Listenable Data Items" on page 184.
di-sortable	Optional	Enter `yes` or `no`, to specify whether this data item should be included in the `Advanced Sort` window of the Supervisor Work Area. The default value is `no`. For further information about the `Advanced Sort` window, see the "Supervisor" section of Genesys Desktop online help.
di-filterable	Optional	Enter `yes` or `no`, to specify whether the data item should be included in the `Filter` and `Find` windows of the Supervisor Work Area. The default value is `no`. For further information about the `Filter` and `Find` windows, see the Supervisor section of Genesys Desktop online help.

Example XML Data Item Definitions

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<dataitems>
    <dataitem>
        <di-fieldID>SalesNA</di-fieldID>
        <di-short-name>Sales NA</di-short-name>
        <di-full-name>Total Sales to North America</di-full-name>
        <di-resource-type>Person</di-resource-type>
        <di-data-type>Float</di-data-type>
        <di
webservice>http://ThirdPartyDataServer.abc.com:8080/ThirdPartyDataA
pp</di-webservice>
        <di-keyfield>EmployeeID</di-keyfield>
        <di-listenable>yes</di-listenable>
        <di-sortable>yes</di-sortable>
        <di-filterable>yes</di-filterable>
    </dataitem>
    <dataitem>
        <di-fieldID>SalesEU</di-fieldID>
        <di-short-name>Sales EU</di-short-name>
        <di-full-name>Total Sales to European Union</di-full-name>
        <di-resource-type>Person</di-resource-type>
        <di-data-type>Float</di-data-type>
        <di-
webservice>http://ThirdPartyDataServer.abc.com:8080/ThirdPartyDataA
pp</di-webservice>
        <di-keyfield>EmployeeID</di-keyfield>
        <di-listenable>yes</di-listenable>
        <di-sortable>yes</di-sortable>
        <di-filterable>yes</di-filterable>
    </dataitem>
    <dataitem>
        <di-fieldID>Tier</di-fieldID>
        <di-short-name>Tier</di-short-name>
        <di-full-name>Performance Tier</di-full-name>
        <di-resource-type>Person</di-resource-type>
        <di-data-type>String</di-data-type>
        <di-
webservice>http://ThirdPartyDataServer.abc.com:8080/ThirdPartyDataA
pp</di-webservice>
        <di-keyfield>FirstName</di-keyfield>
        <di-keyfield>LastName</di-keyfield>
        <di-listenable>no</di-listenable>
        <di-sortable>yes</di-sortable>
        <di-filterable>yes</di-filterable>
    </dataitem>
</dataitems>
```

Adding Data Items to Genesys Desktop Views

You can use the View Template Wizard to create views that include the data items. Each data item will be included with the standard resource properties. The attribute `di-short-name` defines the `Short Name`, and the attribute `di-full-name` defines the `Full Name`. For further information about the View Template Wizard, see the "Supervisor" section of your online help in the Genesys Desktop.

Summary of Actions

All requests between the Genesys Desktop web service and your web service must include an action parameter (see Table 21). Action parameters tell the respective web service which function to perform.

Table 21: Adding Data Items to Genesys Desktop Views

Action	Required Parameters	Purpose	Direction
setListener	• `listenerID` • `fieldID` • keyfield-value pairs • URL of the Genesys Desktop web service	Instructs your web service to send events to the Genesys Desktop web service for the specified data item. The `listenerID` parameter identifies the Genesys resource, the `fieldID` parameter identifies the data item, and the keyfield-value pairs are used to identify the corresponding resource in your system.	Genesys Desktop web service to your web service.
setData	• `listenerID` • `fieldID`-value pairs	Sent to the Genesys Desktop web service by your web service, to set the value of one or more data items. The `listenerID` parameter identifies the Genesys resource, and the `fieldID`-value pairs include the data item values.	Your web service to Genesys Desktop web service.
removeListener	• `listenerID` • `fieldID`	Instructs your web service to remove a listener created by a `setListener` request. The `listenerID` parameter identifies the Genesys resource, and the `fieldID` parameter identifies the data item.	Genesys Desktop web service to your web service.

Table 21: Adding Data Items to Genesys Desktop Views (Continued)

Action	Required Parameters	Purpose	Direction
getData	• fieldID • keyfield-value pairs	Requests that your web service return the value of the data item. The fieldID parameter identifies the data item, and the keyfield-value pairs are used to identify the resource in your system.	Genesys Desktop web service to your web service.

Actions for Listenable Data Items

Adding actions for listenable items to Genesys Supervisor Desktop views is described in "Summary of Actions" on page 183. This section provides further details about these actions.

setListener

If a data item is defined as listenable, the Genesys Desktop web service sends a setListener request to your web service. The request includes the following information:

- listenerID that identifies the Genesys resource. When using the setData action to update the data item(s), you must specify the listenerID.
- fieldID of the data item.
- Keyfield(s) used to identify the corresponding resource in your system.
- URL of the Genesys Desktop web service that will receive the events.

The keyfields must be sufficient for your web service to identify the resource in your system. For example, when working with Person resources, you might want to use the keyfield EmployeeID. For a complete listing of the keyfields available for each type of Genesys resource, see the description of the di-keyfield attribute, page 181

Notes: The resource type is always included in the setListener request. For a complete listing of the resource types, see "di-resource-type" on page 180.

Your web service must send a setData request after receiving the setListener request, and each time the value of the data item changes. For further information, see the "setData" section.

setData

If a data item is defined as listenable, your web service must send a setData request to update the value in the Genesys Desktop web service. The request must include the listenerID and one or more fieldID-value pairs. After the Genesys Desktop web service receives the data, the change(s) are reflected immediately in the Supervisor Work Area views. The Genesys Desktop web

service also sends one of the following response-code values to your web service:

- `Success`
- `Incorrect Data Type`
- `Resource Not Found`
- `No Listener ID`

removeListener

When the Genesys Desktop web service no longer requires real-time events for a data item, it issues a `removeListener` request in the same general manner that it issued the `setListener` request. The request includes the `listenerID` and the `fieldID`.

Listenable Data Items Request Flows

Table 22 shows request flows for listenable data items.

Table 22: Request Flows

Genesys Desktop Web Services	Your Web Service
Issues `setListener` command, passing `listenerID`, `fieldID`, keyfield-value pairs, and URL of the Genesys Desktop web service.	Adds `listenerID` to table, indexed by keyfields.
Sets data in appropriate resource. Each change is immediately reflected in Supervisor Work Area views.	Issues `setData` command, passing `listenerID` and `fieldID`-value pairs.
Issues `removeListener` request, passing `listenerID` and `fieldID`.	Removes listener from listener table.

Example HTTP Requests

Genesys Desktop Web Service to Your Web Service

- `HTTP://ThirdPartyDataServer.abc.com:8080/ThirdPartyDataApp?action=setListener&listenerID=101&fieldID=SalesNA&ResourceType=Person&EmployeeID=112233&webService=HTTP://GenesysDesktopServer.abc.com:8080/AccessData`
- `HTTP://ThirdPartyDataServer.abc.com:8080/ThirdPartyDataApp?action=removeListener&listenerID=101&fieldID=SalesNA`

Your Web Service to Genesys Desktop Web Service

- `HTTP://GenesysDesktopServer.abc.com:8080/AccessData?action=setData&listenerID=101&SalesNA=1250.00`

Actions for Non-Listenable Data Items

Adding actions for non-listenable items to Genesys Supervisor Desktop views is described in "Summary of Actions" on page 183. This section provides further details about using the `getData` request.

getData

If a data item is not defined as listenable, Genesys Desktop obtains the value via a `getData` request. The request contains the `fieldID` of the data item and keyfield-value pairs used to identify the resource in your system. Your web service then returns the current value of the data item in the HTTP response.

Note: The resource type is always included in the `getData` request. For a complete listing of the resource types, see "di-resource-type" on page 180.

Non-listenable Data Item Request Flows

Table 23 shows request flows for non-listenable data items.

Table 23: Request Flows

Genesys Desktop Web Services	Your Web Service
Issues `getData` request passing `fieldID` and keyfield-value pairs.	Returns value of requested data item.

Example HTTP Request

Genesys Desktop Web Service to Your Web Service

- `HTTP://ThirdPartyDataServer.abc.com:8080/ThirdPartyDataApp?action=getData&fieldID=Tier&ResourceType=Person&FirstName=Joe&LastName=Smith`

Supervisor Hierarchy Customization Capabilities

The supervisor desktop views are rendered using a display engine that enables several points where customization can be introduced; for example:

1. The drop-down list enables you to select a category of contact center objects, such as `Agents` and `Agent Groups`. Each entry within a drop-down list is called an object set. Object sets have a visual containment hierarchy that is reflected in the drop-down list generated from them. The object set hierarchy is specified in an eXtended Markup Language (XML) file. Object sets build the menu hierarchy by containing other object sets.

2. Each object set has an associated view set. A view set describes the information that is displayed by the view when the object set is selected, and also indicates a process for rendering the view. The rendering process is done by a Java Server Page (JSP) file.

The Hierarchy Customization feature enables you to include additional entries in the Supervisor Hierarchy drop-down menu by pointing to external web pages. External web pages open directly in the Supervisor Work Area. Both primary menu and sub-menu items are supported. Assigned custom view profiles enable each supervisor to see their own subset of all primary level menu items.

The hierarchy customization feature consists of two related functional parts, described in the following sections:

- “Adding additional entries to the Supervisor Hierarchy drop-down menu” on page 187.
- “Managing the visibility of drop-down menu items using view profile templates” on page 188

Adding additional entries to the Supervisor Hierarchy drop-down menu

The drop-down list description is located in the following XML file:

`supervisor\xml\objectsets.xml`

You can define additional menu entries by adding nested `<objectset>` nodes of `resource_type 1000 (TYPE_WEB_PAGE_GROUP)` and `resource_type 1001 (TYPE_WEB_PAGE)` in the hierarchy of the object sets. See the following example:

```
<objectset id="home" resource_type='1000' title="Company A"
viewset="/supervisor/xml/osets.xml" bundle='none' >
   <objectset id='sales' resource_type='1001' title='Sales'web_page="http://company.com/sales"
      viewset="/supervisor/xml/custom.xml" bundle='none' />
   <objectset id='support' resource_type='1001' title='Technical Support'
    web_page="http://company.com/support"
      viewset="/supervisor/xml/custom.xml" bundle='none' />
   <objectset id='hr' resource_type='1001' title='Human Resources' web_page="http://company.com/hr"
      viewset="/supervisor/xml/custom.xml" bundle='none' />
</objectset>
```

This example creates a drop down with the following hierarchy:

```
Company A
   Sales
   Technical Support
   Human Resources
```

The `id`, `resource type`, and the `viewset` attributes are required.

You must assign a unique ID to each object set that you add.

Object sets with Resource type `1000` should have the predefined viewset: `"/supervisor/xml/osets.xml"`.

Object sets with Resource type `1001` should have the predefined viewset `"/supervisor/xml/custom.xml"`.

The `web_page` attribute contains the full address of the external web page to be opened. This attribute can be omitted if the menu item is for organizing the hierarchy in the menu. Clicking on a menu item that does not have a `set web_page` attribute, does not cause any actions to occur in the Genesys Supervisor Desktop interface.

Figure 46 displays the result of inserting the sample object set descriptions into the `objectsets.xml` file below the `tenants id` object set.

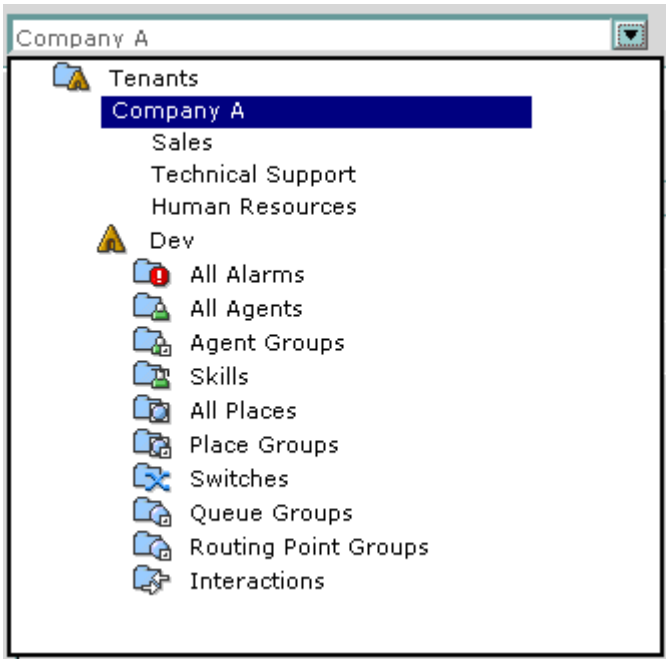


Figure 46: Customized hierarchy. In this example, the object set with the 'home' ID is currently selected and the drop-down has been clicked.

Managing the visibility of drop-down menu items using view profile templates

A View Profile Template may be assigned to any supervisor, enabling the supervisor to see only a subset of all object sets in the hierarchy drop-down menu.

The View Profile Template controls the visibility of the customized entries of the `Supervisor Hierarchy` drop-down menu. View profile templates are created and managed with the View Profile Template Wizard. You can specify which menu items are visible using the current View Profile Template, by checking the corresponding check boxes. If the primary level menu item is not specified

as visible, then all of its children (if any) are also not visible. This information is added to the View Profile Template, which is stored in Configuration Server.

Object sets, representing external web pages, are loaded and treated the same way as other object sets. The View Profile Template is applied when the Supervisor Hierarchy drop-down menu is built, so each supervisor will see a customized supervisor hierarchy menu. If no View Profile Template is assigned to a supervisor, then the full supervisor hierarchy menu, without exclusions, is displayed.

Customization for Open Media View

Genesys Desktop Supervisor 7.6 allows you to create customized views for Open Media interactions according to their media types, for example, sms, fax, e-mail, voice, chat, and so on.

This is possible by adding view templates to the following file:

`/gdesktop/supervisor/xml/emails.xml`

For each view template, specify a correct unique ID and use localized keys for sentences, text labels, and also IDs of `dataitem` elements presented in the following steps.

1. Add a new view to this file by creating a view template: `<viewtemplate id="..." title="..." customizer="..." />`.

   Note: The `id`, `title`, and `customizer` attributes are required.

2. To specify that the view template targets Open Media interactions, you should fill in the attribute `mediaType` with one Open Media media type, or with a list of Open Media media types separated by commas.

   `mediaType="sms,fax"`

   If you do not specify a value for the `mediaType` attribute, the default value is `email`.
   If you specify a list of media types for the `mediaType` attribute, the view template is added to the view of any interaction whose media type belongs to this list.

   Note: If you add the parameter `mediaType` to former view templates, these view templates appear only for interactions matching the specified media types.

3. Set the `customizer` attribute to point to the customized page that renders properties enclosed by the `viewtemplate` tag. Genesys Desktop Supervisor includes a generic render page `doGenericBody.jsp`.

customizer="doGenericBody.jsp"
The format and external representation of most data items and properties are already defined in project classes.

4. For new view templates, you can specify one or several propertygroup elements that enclose a list of dataitem elements. <propertygroup id="...", title="...">
<dataitem id="..." title="..."/>
</propertygroup>

   Each propertygroup tag requires the id and title attributes. The title attribute specifies a key for the localization of a string in the resource file. This string appears in the header of tab page.
   Each dataitem element describes a property of the interaction resource object. These properties are defined in DataItemDefinitions class, except the propertyLink property, which is a predefined custom property for fax or scanned documents.

View Template for Fax

The following XML code presents a view template for fax Open Media interactions:

```
<viewtemplate id="scan_detail" title="scan_detail.title"
   customizer="doGenericBody.jsp" mediaType="fax">
   <propertygroup id="dets" title="scan_detail.title">
   <!-- Each propertygroup object has an optional customizer
      attribute which use a custom JSP file to build just that piece
      instead of the whole pane -->
   <!-- Specify a list of dataitems to be listened when an
      event flows; the JSP must hardcode the display of dataItems -->
      <dataitem id="i_ReceivedAt" title=""/>
      <dataitem id="i_EmailSubject" title=""/>
      <dataitem id="propertyLink" title=""/>
     </propertygroup>
</viewtemplate>
```

Chapter

8 Rebranding Genesys Desktop for OEMs

This chapter presents the technical details that original equipment manufacturer (OEM) developers require to rebrand Genesys Desktop successfully. The topics in this chapter include:

Labels

The Genesys Desktop graphical interface contains several labels that display the word "Genesys" or terms that are specific to the Genesys brand.

These labels reside in the OEM file `brand.properties`, which is located here:

```
[GD Install Path]\webapps\gdesktop\web-inf\classes\com\genesyslab\uadthin\oem\
```

The structure of the Genesys-specific labels in `brand.properties` is:

- company-name-long=*Company Name*
- product-name=*Product Name*
- about-warning=*Legal Disclaimer*

Make your changes on the right-hand side of the equal sign (=).

Bitmaps

The Genesys Desktop graphical interface contains several bitmaps that display the Genesys word mark and/or logo. Two OEM images are used to replace them:

- `logo.png`: This file is displayed in the screen that replaces the login page, after the login is submitted.
- `idc_logo.gif`: This file is displayed in the `About` box for the logo. The dimensions of this image are 269 pixels by 98 pixels.

The OEM version of these files must be located here:
`[GD Install Path]\webapps\gdesktop\oem`

Help

Genesys Desktop employs a set of HTML files to display online help. There is one subset for Genesys Agent Desktop (GAD) and one subset for Genesys Supervisor Desktop (GSD). There is one file subset for each localized version of Genesys Desktop. The subsets are managed in a hierarchy of folders (see "OEM Online Help File Hierarchy").

The entry points for OEM HTML pages are:

- Agent only: `gad\<Language>\AgentDesktop.htm`
- Agent and supervisor: `gsd\<Language>\76DesktopHelp.htm`

OEM Online Help File Hierarchy

The OEM online help file hierarchy is in this format:

```
[GD Install Path]\webapps\gdesktop\oem\
  help\
    gad\
      def\ [contains all online help files for GAD in the default language]
      fr_FR\ [contains all online help files for GAD in French]
      ...
    gsd\
      def\ [contains all online help files for GSD in the default language]
      fr_FR\ [contains all on line help files for GSD in French]
      ...
```

Index

G

H

I

J

K

L

M

N

O

P

R

S

T

V

W

X