Workspace Desktop Edition Developer's Guide

Introduction

4/9/2025

Introduction

Purpose: Presents the architecture and design concepts of the Interaction Workspace.

Contents

Introducing Interaction Workspace

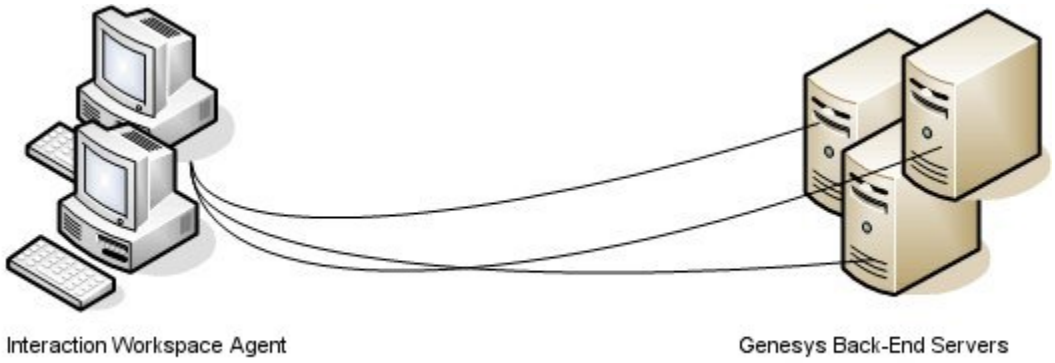
Interaction Workspace is the customer interaction interface for the Genesys 8 software suite. Interaction Workspace contains many components that you can use to enrich the content of Interaction Workspace with custom views and functionality. You can use the following Interaction Workspace components to customize your interface:

- **Platform SDK**-The low-level SDK that is used to access Genesys back-end servers
- **Enterprise SDK**-The high-level SDK that is built on top of Platform SDK and is used to render models and services
- **Interaction Workspace API**-The methods that are used to implement extensions for Interaction Workspace

A set of Interaction Workspace Extension Samples is also provided to illustrate the best coding practices for Interaction Workspace customization. **Limitation**: Usage of Enterprise SDK that is provided with this release of Interaction Workspace is supported only for the purpose of Interaction Workspace customization.

Architecture

The following figure illustrates a minimal deployment that consists of agent workstations that are connected directly to the Genesys back-end servers.



Simple Client-Server Architecture

Dependency Injection Container Application Block

The Dependency Injection Container Application Block is available for use when you use loosely-coupled applications to develop Interaction Workspace. This lightweight, extensible Dependency Injection container enables developers to build loosely-coupled applications and provides the

following advantages:

- Simplified object creation, especially for hierarchical object structures and dependencies
- Abstraction of requirements, enabling developers to specify dependencies at run time or in configuration, and to simplify management of crosscutting concerns
- Increased flexibility by deferring component configuration to the container
- Service location capability, enabling clients to store or cache the container
- Instance and type interception

Note: Genesys Enterprise SDK also implements Dependency Injection recommendations, which makes integration easier.

Technical and Design Concepts	Application to Customization
Dependency Injection and Inversion of Control	Used by developers to declare and retrieve alternative implementation of services, models, views, and presenters. Developers can use the Dependency Injection and Inversion of Control when they are developing software.

Loosely-coupled Application Library

The Interaction Workspace is built by using a loosely-coupled application library. This library is used by developers to create composite Windows Presentation Foundation (WPF) applications. It is designed to help architects and developers achieve the following objectives:

- Create a complex WPF application from modules that can be built, assembled, and optionally, deployed by independent teams.
- Minimize cross-team dependencies and enable teams to specialize in different areas, such as UI design, business logic implementation, and infrastructure code development.
- Use an architecture that promotes reusability across independent teams.
- Increase the quality of applications by abstracting common services that are available to all the teams.
- Incrementally integrate new capabilities.

The Interaction Workspace provides guidance and implements patterns that make customization easier. In general, all of these concepts have a common aim, which is to implement loosely-coupled applications and ease extensibility.

Technical and Design Concepts	Description	Application to Customization
Model View ViewModel (MVVM)	The `MVVM` can separate the responsibilities of the visual display and the responsibilities of user interface state and behavior into different classes named `View` and `View Model`, respectively.	You can build an alternate custom `View` for any given out-of-the-box `ViewModel`, and you can build an alternate custom `ViewModel` for any given out-of-the-box View.

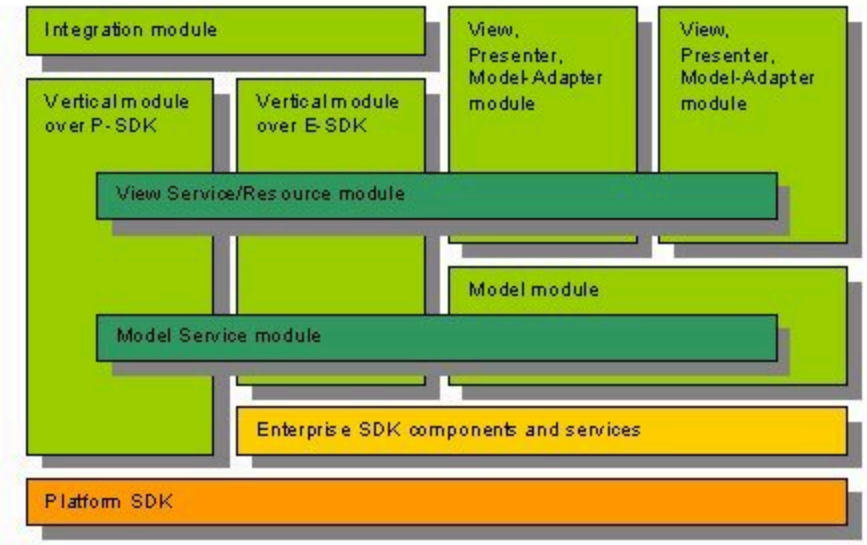
Technical and Design Concepts	Description	Application to Customization

	• The `View` class manages the controls on the user interface.
	• As a *facade* on the model, the `View Model` class provides you with UI-specific state and behavior:
	  • It encapsulates the access to the model.
	  • Its public interface is easy to consume from the `View` (for example, for using data binding).	

Module	A module can be individually developed, tested, and deployed by different teams.	Customization can be implemented by partners or by customers, in a reusable or single-use purpose.
Region Manager	Regions enable a compositional pattern and are commonly used in template layouts and multiple view layouts.	Integration of custom views into out-of-the-box named and documented Regions is simplified, even without knowledge of the application construction. For example, in a typical application, a region can be a tab area.

Interaction Workspace Modules

As defined in the software development kit (SDK), a module is a software element that can be individually developed, tested, and deployed by different teams. Interaction Workspace contains several modules that can cover one or several layers of the application. They can contain views, presenters (ViewModel), or models. In general, the functional modules cover the full stack, whereas the service modules focus on a particular layer. By using a modular approach in the application, Genesys is able to provide an SDK to developers who are planning to add customized code into Interaction Workspace. The figure below summaries the various types of modules in Interaction Workspace.



Types of Modules in Interaction Workspace

<references/>

Read Next

Introducing Extensions