Siebel eScripts code assist

This whitepaper provides a comprehensive overview of how Siebel eScript Code Assist enhances Siebel CRM development. We examine the tool that synchronizes workspaces, download Business objects, delivers AI-driven suggestions and enhances Siebel Object Server Scripts development as a Visual Studio Code plugin.

November, 2025, Version [1.0]

Purpose statement

This document provides a solution approach and is for informational purposes only. It is intended solely to provide valuable insights and guidance on the subject matter while serving as a reference for stakeholders. This document does not constitute a binding agreement or official policy.

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement, nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the solution described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Table of contents

Overview

Siebel CRM (Customer Relationship Management), is a comprehensive suite of applications designed to help organizations manage and automate their customer-facing processes, spanning sales, marketing, and customer service. Recognized for its robust nature and adaptability, Siebel provides a unified, 360-degree view of the customer, enabling businesses across various industries to streamline operations, enhance customer engagement, and ultimately drive revenue growth. It is a highly configurable platform that supports diverse deployment models, accommodating the complex needs of large enterprises.

Siebel eScript is the most popular scripting language used by developers to extend and customize Siebel CRM functionality beyond its core declarative configuration capabilities. Based on the ECMA-262 standard (which forms the core of JavaScript), eScript provides a powerful mechanism to implement complex business logic, perform data validation, manage user interactions, and facilitate integration with external systems. Developers use eScript to create both server scripts (executed on the Siebel Server) and browser scripts (executed in the client's web browser), allowing for precise control over the application's behavior and user experience.

Siebel eScript Code Assist is a Visual Studio Code extension that enables seamless editing and AI-assisted development of Siebel Object Server Scripts directly within VS Code. It leverages Siebel REST APIs for real-time synchronization and integrates OCI Generative AI to provide intelligent code suggestions and assistance. The built-in AI Assistant offers contextual recommendations based on the current eScript file and selected code line. This solution is developed on top of the Endoit (https://endoit.hu/innovates/scripteditor/) .

Solution Overview

Optimized specifically for **Siebel eScript**, this powerful new plugin seamlessly integrates with your existing workflow via **Microsoft Visual Studio Code**. We are releasing a **beta version** of the tool that you can download today. It's designed for **fast setup and easy enhancement**, providing a foundational code base that allows your team to **customize and build new capabilities** tailored precisely to your organization's unique development and compliance requirements.

This initial beta release is ready to transform your development cycle with the following key features:

- **Synchronize Servers**: Seamlessly pull or deploy Siebel business object eScripts by connecting to a designated Siebel Server.
- **Accelerate Coding:** Generate robust code snippets to dramatically boost developer productivity.
- **Strengthen Quality:** Automatically create **unit tests** to achieve higher code coverage and reliability.
- **Improve Readability: Annotate code** quickly, making it easier to maintain and onboard new developers.
- **Fast-Track Learning: Explain complex code sections** on demand, helping developers accelerate their understanding of existing logic.
- **Instant Support:** Get **chat-based answers** to your critical, coding-related questions right within the editor.

Code Assist architecture

The architecture of **Siebel Code Assist** is elegantly designed around two key components: a **Visual Studio Code plugin** and a powerful **API-connected backend service.** The plugin acts as your development interface, allowing you to pull Siebel workspaces directly into VS Code locally, access context-aware help via a chat interface, and push completed script and web template changes back to Siebel. Meanwhile, the backend does the heavy lifting, running optimized **Large Language Models (LLMs)** on **OCI GPUs** and utilizing a **streaming interface** to ensure low latency and responsive, real-time code generation suggestions.

Privacy and data collection

Code Assist never stores any code snippets that you provide in your prompts. Your prompt and any code provided is only employed temporarily by the backend service until a satisfactory response is provided. When Code Assist provides the response, your prompt is deleted.

Likewise, Code Assist doesn’t collect personally identifiable information (PII). Oracle understands and respects that customer privacy is important to you.

Get started

The Bottom Line: Faster, Smarter Siebel

Siebel Code Assist isn't just an add-on; it's the future of efficient Siebel customization. By offloading complex code generation to powerful **cloud-based LLMs** and seamlessly integrating the output directly into **VS Code**, Oracle has eliminated the friction traditionally associated with scripting. This robust two-part architecture means your developers can stop wrestling with syntax and start focusing on **delivering business value faster** than ever before. It’s time to move beyond the manual workflow and embrace the productivity leap offered by intelligent assistance.

Use Case Deep Dive

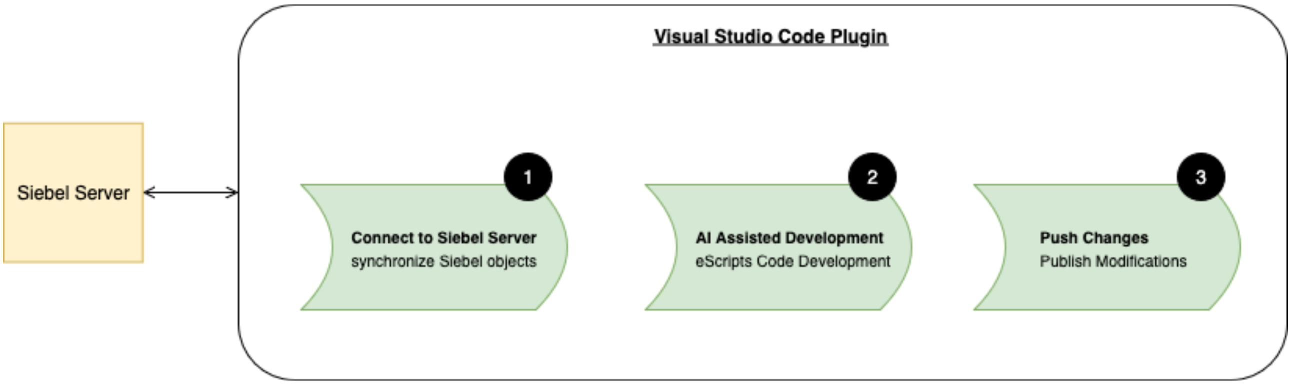


Figure 1: Demo Flow

1. **Connect to Siebel Server and Manage Workspaces** - Establish and save secure connections to specific Siebel Servers, enabling you to search, browse, and download Siebel objects—including Business Services, Business Components, Applets, Applications, and Web Templates—directly to your local development environment. Real-time synchronization via Siebel REST APIs ensures your workspace stays aligned with the server, and multiple workspaces can be managed efficiently.

2. **AI-Assisted eScript Development** – Leverage a VS Code-integrated chat interface with automatic context updates for files or selected lines. Receive AI-generated, Siebel-efficient code suggestions, explanations for complex logic, and interactive guidance without switching tools. Integration with OCI Generative AI further enhances productivity, simplifies development and accelerates learning for developers.

3. **Publish Modifications** – Seamlessly publish all changes made in the local environment back to the Siebel Server, ensuring updates are deployed accurately and efficiently. This feature enables developers to maintain synchronization between local development and the live environment, supporting consistent and reliable application updates.

Technical Diagram

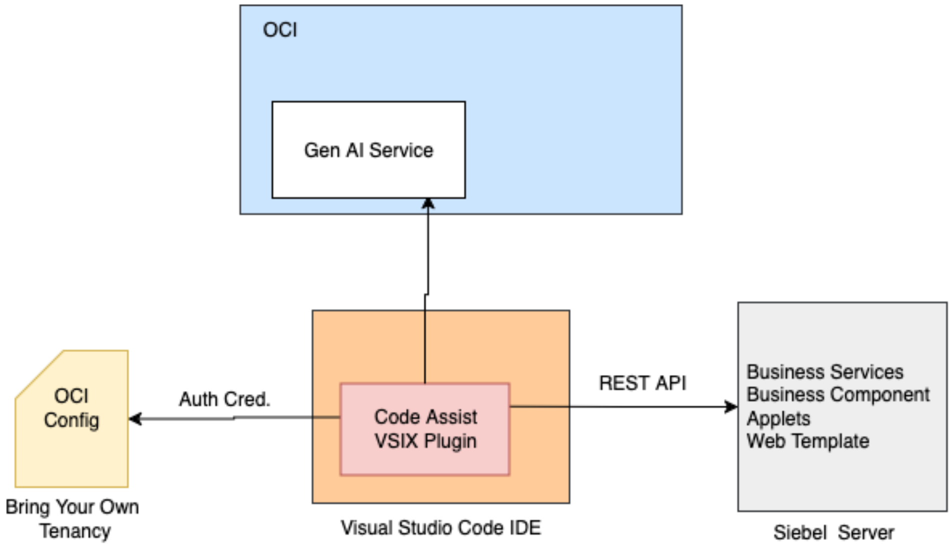


Figure 2: Architecture diagram

Technical Deep Dive: Siebel eScripts Code Assist

This section details the technical architecture and component interactions that enable AI-assisted development within the Siebel ecosystem. At the center of this solution is the Siebel eScripts Code Assist VS Code plugin, which integrates local developer tooling with Siebel Server REST APIs and OCI Generative AI services to accelerate coding, enhance quality, and streamline workspace management.

Architecture Overview

The architecture is composed of three primary components: the Developer IDE, the Siebel Server Services and the OCI AI Services.

- **Developer IDE**: This layer represents the developer's working environment, The Visual Studio Code IDE enhanced with the Code Assist VSIX plugin. It manages Siebel workspace connections, retrieves and stores Siebel objects locally, and keeps the development environment synchronized with server-side changes. The VS Code plugin also handles OCI authentication (using Bring Your Own Tenancy configurations from `~/.oci`) and provides the chat-driven UI used for AI-powered code assistance.

- **Siebel Server Services**: This layer encompasses the Siebel business logic and metadata services exposed through Siebel REST APIs. Using these APIs, the plugin can fetch or update Siebel artifacts such as Business Services, Business Components, Applets, Applications, and Web Templates. These server-side components serve as the authoritative source for object definitions and customizations, ensuring the local workspace remains aligned with active Siebel workspaces.

- **OCI AI Services**: This layer provides the intelligence that powers the Code Assist experience, which process context retrieved from the developer's current file or selected lines in VS Code. These services generate high-quality eScript code snippets, explain complex logic, and offer targeted suggestions to improve clarity, maintainability, and performance. Developers authenticate to OCI using their own tenancy configuration, ensuring isolation, security, and enterprise compliance.

Technical Significance

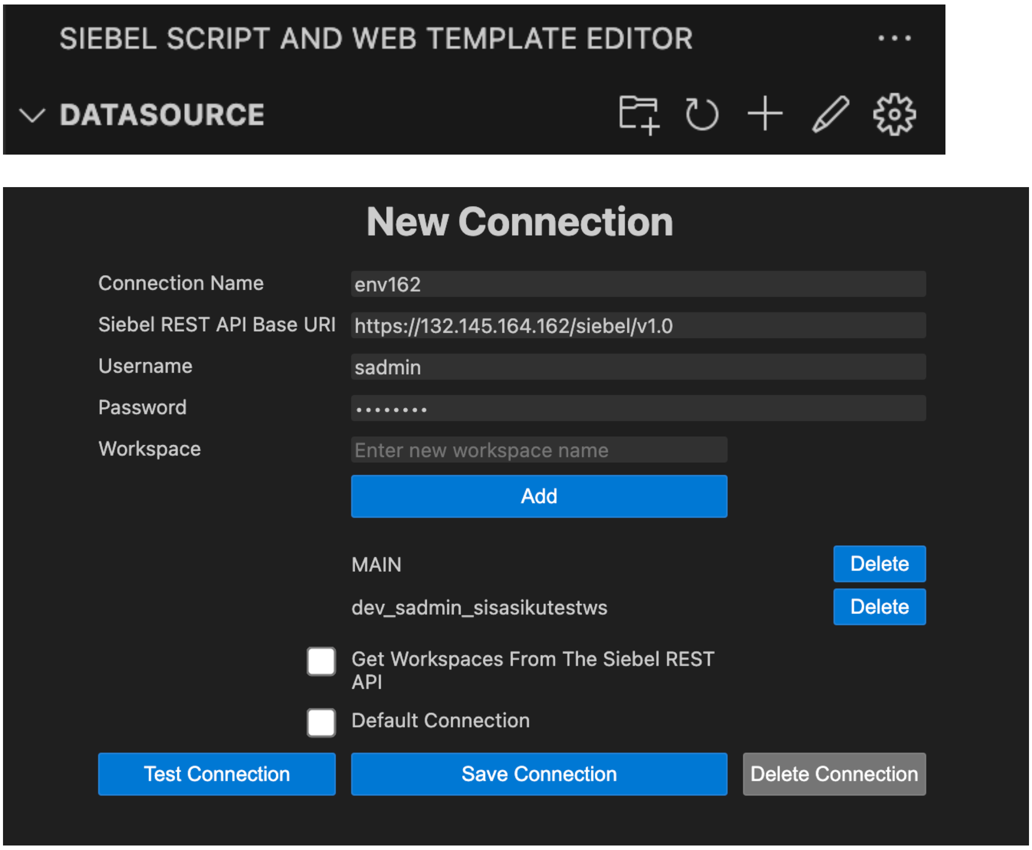
The integration pattern demonstrates a modern, decoupled approach to enhancing Siebel development workflows without altering core Siebel architecture. By using Siebel REST APIs as the gateway to server-managed objects, the system preserves Siebel's native metadata model while extending capabilities externally through VS Code. This separation of concerns enables rapid innovation: the Code Assist AI engine can evolve independently, leveraging new LLMs, improved context handling, or advanced code analysis while remaining fully compatible with existing Siebel deployments.

Prerequisites

- Download Visual Studio Code IDE
- Open Visual Studio Code, navigate to the Extensions panel, click the three-dot menu, and choose "Install from VSIX…". Select the file: siebelscriptandwebtempeditor-2.2.0.vsix to install the Code Assist plugin.
- Bring Your Own Tenancy: Place your OCI configuration file including the private key into the local ~/.oci directory to enable secure access to OCI Generative AI services.

Configuration

1. To open the extension, click on the new icon in the left sidebar. To use the extension, at least one Siebel Connection must be configured. Click on the **New Connection (+)** button in the Datasource panel, and a dedicated GUI opens:



Connection Name: env162

Siebel REST API Base URI: https://132.145.164.162/siebel/v1.0

Username: sadmin

Password: ••••••••

Workspace:

MAIN

dev_sadmin_sisasikutestws

□ Get Workspaces From The Siebel REST API

□ Default Connection

Figure 3: Connection setup

2. Fill the Env Details:
    - `Connection Name` is a unique name to identify the connection. It is the name displayed in the extension's UI when selecting the connection. Use something that identifies the connection for you easily, such as SANDBOX, DEV, INT, etc. This cannot be changed later.
    - `Siebel REST API Base URI` defines access to the Siebel REST API in the following format: *`https://Server Name:Port/siebel/v1.0`*, where:
        - `Server Name` is the URL of the Siebel server.
        - `Port` is the HTTP port where the Siebel listens for connections.
        - e.g. if the Siebel Server URL is *`https://dev.testserver.local`*, and the port is `443`, then it shall be *`https://dev.testserver.local:443/siebel/v1.0`*.
    - `Username` is the username used to access the Siebel with basic authentication.
    - `Password` is the password used to access the Siebel with basic authentication with `Username`.
    - Use the **Test Connection** button to test if the connection is working, an information message is shown if it is ok, or an error message if something is amiss. Then save with the **Save Connection** button, and the panel changes to the Edit Connection GUI, with three more settings appearing:
3. Note: If there is no valid certificate installed for a Siebel connection, the REST API calls result in the following error: Error using the Siebel REST API: self signed certificate in certificate chain. The recommended solution is to get and install a valid certificate for the given Siebel endpoint, other possibility is to set the following environment variable, **WARNING: THIS IS HIGHLY DISCOURAGED FOR SECURITY REASONS**:

```
NODE_TLS_REJECT_UNAUTHORIZED = 0
```

Settings

Settings can be opened with **Settings (cogwheel)** button from the Datasource panel:

- **Connections**: contains the connection data, do not modify this in the settings.json. When updating the extension from older (< 2.0.0) versions, the deprecated settings related to connection and workspace configurations are migrated here in suitable format.
- **Default Connection**: name of the default connection, do not modify here, set using the **Default Connection** checkbox on the Edit Connection GUI.

The other settings customize the behavior of the extension. Changes in them are reflected immediately:

- **Default Script Fetching**: default method for fetching server scripts when clicking on a Siebel object:
    - None - always ask: always asks how to fetch (default).
    - Only method names: fetches only the script names of the Siebel object.
    - All scripts: always downloads all scripts of the Siebel object. The behavior of already downloaded files depends on the **Default Action When File Exists** setting.
- **Local File Extension**: the default file extension for the downloaded server scripts, .js (default) or .ts. If you use explicit types in your eScript scripts, .ts is recommended, else errors are shown for using types in a .js file.
- **Max Page Size**: number of records to fetch from the Siebel REST API. Cannot be greater than the value of the MaximumPageSize parameter of the component EAI Object Manager (default is 100, in Siebel and in the extension as well), otherwise Siebel responds with an error.
- **Single File Auto Download**: if this setting is true (default is false), scripts and web templates are downloaded without asking for permission when clicking on them. The behavior when clicking on already downloaded files depends on the **Default Action When File Exists**.
- **Default Action When File Exists**: the default action when selecting an already downloaded script or web template from the tree views:
    - None - always ask: always asks what to do, if multiple scripts are downloaded, then it is asked for each existing file (default).
    - Open file: opens the file from the disk, which can be refeshed manually with the pull button.
    - Overwrite: pull the content from Siebel before opening it. **WARNING**: this overwrites files which are already downloaded!

User interface

The UI consists of six different panels, the uppermost is used for selecting the connection and workspace from where the data is fetched, the type of the Siebel object and a search bar: This panel has three or four buttons in its title bar:

- **Refresh Workspaces (refresh)**: only shown when the **Get Workspaces From The Siebel REST API** is true, synchronizes the workspaces with Siebel, which is useful if new workspaces were created, their status changed or a connection error occured
- **New Connection (+)**: opens the New Connection GUI to create a new connection.
- **Edit Connection (pencil)**: opens the Edit Connection GUI to edit the selected connection, if there are no connections, it opens the New Connection GUI as well.
- **Open Settings (cogwheel)**: opens the settings of the Siebel Script And Web Template Editor extension.

The other five panels display the list of Siebel objects (Business Services, Business Components, Applets, Applications and Web Templates) and for the first four, their respective server scripts, and they have Collapse All button too, which collapses the opened scripts in the tree view. Individual panels can be hidden from the top Views and More Actions menu (...) menu.To get Siebel server scripts or web templates, select a connection, a workspace and an object type. Start typing the name of sought object into the search field, wildcard character * can be used (is appended automatically to the end of the search string).

Getting server scripts from Siebel

Click on an object in the list in a panel, and if the **Default Script Fetching** setting is None - always ask, a dialog box opens in the bottom right corner with three buttons, else one of its first two options happen instead:

- **Yes**: gets and downloads all server scripts for the object (same as the All scripts option).
- **Only method names**: gets only the method names (same as the Only method names option) as children of the object in the list, which can be downloaded individually by clicking on their names if the **Single File Auto Download** setting is true, otherwise after a confirmation.
- **No**: closes the dialog box.

A checkmark is displayed in front of each object which has at least one method downloaded, and checkmarks in front of script names indicate which ones are already on disk.

Scripts are stored in the first VS Code workspace folder (only one should be open) as javascript or typescript files with .js or .ts extension depending on the **Local File Extension** setting, in the following folder structure:

```
ConnectionName\WorkspaceName\ObjectType\ObjectName\MethodName.js
```

Getting web templates from Siebel

Click on a web template, and depending on the **Single File Auto Download** setting, the web template gets downloaded instantly or after a confirmation. A checkmark is displayed in front of the web templates if they exist on disk. Web templates are stored in the first VS Code workspace folder (only one should be open) as html files with .html extension in the following folder structure:

```
ConnectionName\WorkspaceName\webtemp\WebTemplateName.html
```

Pulling, pushing, comparing and searching in scripts/web templates

The following action buttons are shown in top right corner of the VS Code editor when eligible files are open:

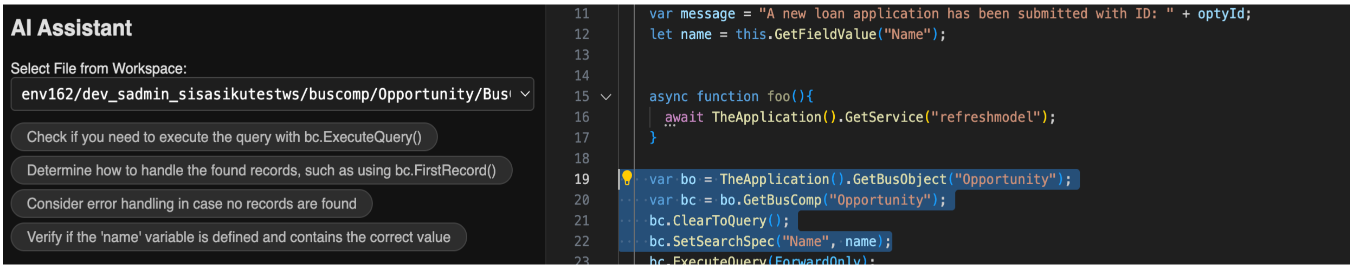
- **Pull (downward pointing chevrons)**: after confirmation, pulls the current object from Siebel to overwrite the file.

- **Push (upward pointing chevrons)**: after confirmation, pushes the current object into Siebel to update it, and a success/error message is shown. If the file name differs from the contained function name, an error is thrown by the extension, only `(declarations)` is exempt from this check. New server scripts can be created by making new files in the appropriate folder with the same file name as the future function name, and having .js or .ts extension. Subsequently pushing them to Siebel create them as new server scripts of the Siebel parent object. This does not work for web templates, an error is thrown if they do not already exist in Siebel. Pushing is only allowed into workspaces created by the username specified in the connection; it is not permitted into the MAIN or integration workspaces (the button will be hidden in those cases).
- **Compare (rectangle with the two-way arrow)**: compares the downloaded object with its versions in other Siebel workspaces. It shows a workspace selection at the top of the screen, either all Siebel workspaces if **Get Workspaces From The Siebel REST API** is enabled, or only the ones present in the connection. After choosing a workspace to compare the current file against, a difference editor is opened, with the object from the choosen Siebel workspace on the left side, and the current file on the right,

  changes can be reverted in the downloaded one:

- **Pull missing and search in scripts (magnifying glass with the squiggly line)**: visible only for scripts, fetches all missing scripts for the current Siebel parent object, and searches in them for the selected text or word in the Search view.

Using AI Code Assist



AI Assistant

Select File from Workspace: env162/dev_sadmin_sisasikutestws/buscomp/Opportunity/Bus

Check if you need to execute the query with bc.ExecuteQuery()
Determine how to handle the found records, such as using bc.FirstRecord()
Consider error handling in case no records are found
Verify if the 'name' variable is defined and contains the correct value

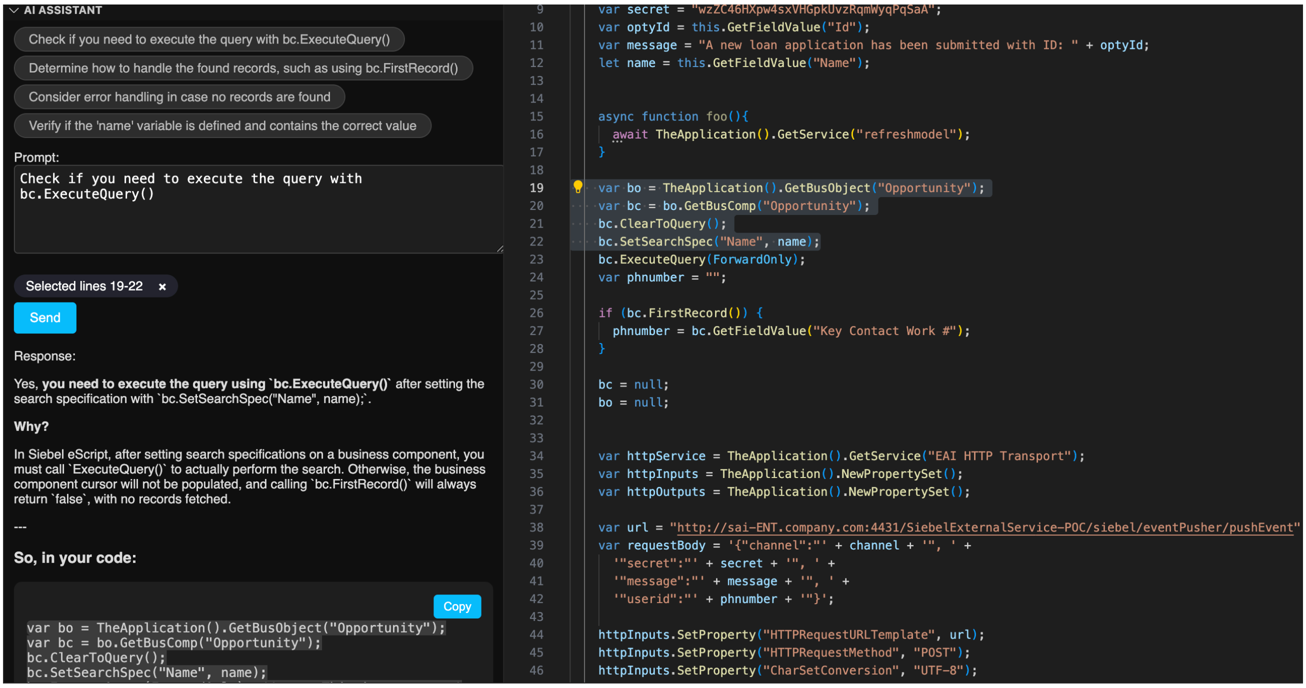
```
var message = "A new loan application has been submitted with ID: " + optyId;
let name = this.GetFieldValue("Name");


async function foo(){
    await TheApplication().GetService("refreshmodel");
}

var bo = TheApplication().GetBusObject("Opportunity");
var bc = bo.GetBusComp("Opportunity");
bc.ClearToQuery();
bc.SetSearchSpec("Name", name);
bc.ExecuteQuery(ForwardOnly);
```

Figure 4: selecting workspace

"**Select File from Workspace**" dynamically updates the file context based on the currently active file in the editor. The AI Assistant intelligently suggests relevant questions when prompted, enabling users to receive contextual assistance powered by OpenAI models hosted on Oracle Cloud Infrastructure (OCI). The Authentication is done via ~/.oci Config file allowing users to bring their own tenancy.



Check if you need to execute the query with bc.ExecuteQuery()

Determine how to handle the found records, such as using bc.FirstRecord()

Consider error handling in case no records are found

Verify if the 'name' variable is defined and contains the correct value

Prompt:

```
Check if you need to execute the query with
bc.ExecuteQuery()
```

Response:

Yes, **you need to execute the query using `bc.ExecuteQuery()`** after setting the search specification with `bc.SetSearchSpec("Name", name);`.

Why?

In Siebel eScript, after setting search specifications on a business component, you must call `ExecuteQuery()` to actually perform the search. Otherwise, the business component cursor will not be populated, and calling `bc.FirstRecord()` will always return `false`, with no records fetched.

So, in your code:

```
var bo = TheApplication().GetBusObject("Opportunity");
var bc = bo.GetBusComp("Opportunity");
bc.ClearToQuery();
bc.SetSearchSpec("Name", name);
```

```
 9  var secret = "wzZC46HXpw4sxVHGpkUvzRqmWyqPqSaA";
10  var optyId = this.GetFieldValue("Id");
11  var message = "A new loan application has been submitted with ID: " + optyId;
12  let name = this.GetFieldValue("Name");
13
14
15  async function foo(){
16    await TheApplication().GetService("refreshmodel");
17  }
18
19  var bo = TheApplication().GetBusObject("Opportunity");
20  var bc = bo.GetBusComp("Opportunity");
21  bc.ClearToQuery();
22  bc.SetSearchSpec("Name", name);
23  bc.ExecuteQuery(ForwardOnly);
24  var phnumber = "";
25
26  if (bc.FirstRecord()) {
27    phnumber = bc.GetFieldValue("Key Contact Work #");
28  }
29
30  bc = null;
31  bo = null;
32
33
34  var httpService = TheApplication().GetService("EAI HTTP Transport");
35  var httpInputs = TheApplication().NewPropertySet();
36  var httpOutputs = TheApplication().NewPropertySet();
37
38  var url = "http://sai-ENT.company.com:4431/SiebelExternalService-POC/siebel/eventPusher/pushEvent"
39  var requestBody = '{"channel":"' + channel + '", ' +
40    '"secret":"' + secret + '", ' +
41    '"message":"' + message + '", ' +
42    '"userid":"' + phnumber + '"}';
43
44  httpInputs.SetProperty("HTTPRequestURLTemplate", url);
45  httpInputs.SetProperty("HTTPRequestMethod", "POST");
46  httpInputs.SetProperty("CharSetConversion", "UTF-8");
```

Figure 5: Chat interface for AI Assistance

The AI Assistant provides intelligent suggestions based on the selected lines and user's query, offering relevant code completions or guidance. Users can conveniently copy the AI-generated code snippets for use directly within their project. The assistance is powered by OpenAI models hosted on Oracle Cloud Infrastructure (OCI).

Autocompletion and semantic checking

Files index.d.ts and jsconfig.json are created in the Siebel VS Code workspace folder if not present already. The index.d.ts contains type definitions and documentation for Siebel specific constants, objects and interfaces, so the Visual Studio Code has greatly improved autocompletion and semantic checking features. Errors are raised if problems are detected, e.g. using non-existing methods on a given Siebel object.

For the best experience, ensure the following setting is set to true in the Visual Studio Code preferences:

- JavaScript > Suggest > Complete Function Calls

Folder structure

Folder structure for the scripts and web templates:

```
Visual Studio Code Workspace folder
├── Connection Name
│   ├── Workspace Name
│   │   ├── service
│   │   │   ├── Business Service Name
│   │   │   │    ├── Service_PreInvokeMethod.ts
│   │   │   │    └── CustomScript1.ts
│   │   │   └── Other Business Service Name
│   │   │        └── CustomScript2.js
│   │   ├── buscomp
│   │   │   └── Business Component Name
│   │   │       ├── Buscomp_PreInvokeMethod.js
│   │   │       └── CustomScript3.js
│   │   ├── applet
│   │   │   └──Applet Name
│   │   │       ├── WebApplet_PreInvokeMethod.ts
│   │   │       └── CustomScript4.ts
│   │   ├── application
│   │   │   └───Application Name
│   │   │       └── CustomScript5.js
│   │   └── webtemp
│   │       └── Custom Web Template.html
│   └── Other Workspace Name
│       ...
├── Other Connection Name
│   └── Another Workspace Name
│        ...
├── compare
│   ├── compare.js
│   ├── compare.ts
│   └── compare.html
├── index.d.ts
└── jsconfig.json
```

Benefits

This initial beta release is ready to transform your development cycle with the following key features:

- **Connect and Save Workspace Connection:** Establish a secure connection to a Siebel Server workspace and save it for easy access in future sessions. This allows you to work seamlessly with the server without repeatedly entering connection details.
- **Pull Siebel Objects to Local Workspace:** Download changes for Siebel objects—including Business Services, Business Components, Applets, Applications, and Web Templates—into your local development environment, ensuring you have the latest versions for editing and customization.
- **Bring Your Own OCI Tenancy:** Use your own OCI configuration (from ~/.oci) to access OCI Generative AI services. This enables AI-assisted code suggestions and ensures secure, enterprise-ready integration with your own cloud tenancy.
- **AI-Assisted eScript Development:** Automatically update the context of the eScript file you're editing and receive intelligent suggestions based on the specific lines selected. This feature helps developers write efficient, high-quality Siebel code and quickly understand complex logic.

- **Push Changes Back to the Workspace:** Publish all updates from your local workspace back to the Siebel Server, keeping your environment synchronized and ensuring that all modifications are accurately deployed.

Future Works

- **Auto inline code completion**: Provides real-time suggestions as you type, enabling faster and more accurate eScript development.
- **Auto lint issues detection:** Automatically identifies coding issues and best-practice violations to improve code quality and maintainability.
- **MCP Server Siebel:** Introduces a dedicated MCP server interface to support richer, more interactive Siebel development workflows.
- **Version Control: Git support:** Integrates seamless Git-based version control to streamline collaboration.

Conclusion

Modern Siebel development is often slowed by manual scripting, repetitive technical tasks, and constant context switching between tools. Siebel eScripts Code Assist fundamentally transforms this workflow. By embedding AI-powered code generation, explanations, and workspace synchronization directly within the familiar Visual Studio Code environment, we're not just adding convenience—we're redefining the developer experience.

Your team can shift from time-consuming boilerplate work to delivering high-quality customizations with greater speed, accuracy, and confidence. This integration is built to make developers more productive, code more consistent, and releases more reliable, all while securely leveraging your own OCI tenancy.

Call To Action

Ready to Transform Your Siebel Development Workflow?

Don't let manual scripting and repetitive technical tasks slow your team down. Take the next step toward smarter, AI-assisted development and unlock a faster, more efficient Siebel build cycle today.

Contact us directly at siebel_coe_grp@oracle.com to discuss your implementation.

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle