Email to Service Request: Automating SRs with Siebel AI

This whitepaper provides a comprehensive overview of how Siebel AI optimizes service request management. We examine the technology that interprets incoming emails, generates and routes SRs to the appropriate experts, and produces detailed closure summaries upon resolution.

October, 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

Table of contents

List of figures and tables

Overview

Siebel CRM, a comprehensive customer relationship management platform, is renowned for its robust and highly customizable solutions for sales, marketing, and customer service. Its email handling feature, often referred to as Siebel Email Response, is a core component that enables businesses to manage high volumes of inbound and outbound email communications with customers. This functionality is built on a scalable architecture that includes dedicated components for receiving, processing, and sending emails.

Managing customer emails is a persistent challenge for many companies. Important messages often go unnoticed or receive incomplete responses, leading to delays and frustration. When a Service Request (SR) is closed, support teams typically spend valuable time manually sorting through long email threads to compile a summary or resolution. Yet, customers expect timely, clear updates without having to ask. The need for an intelligent, automated solution is more pressing than ever—and that's exactly where **Siebel AI** steps in.

Our new GenAI integration is designed to transform agent efficiency. Siebel Email Response users can now, without leaving their familiar workspace, access powerful AI-powered insights, comprehensive summaries, and intelligently drafted replies—all with just a few clicks. Critically, the AI can also generate a precise closure summary that aggregates all Service Request (SR) activities. This feature not only saves agents valuable time but also ensures customers receive a clear, concise overview of everything that transpired.

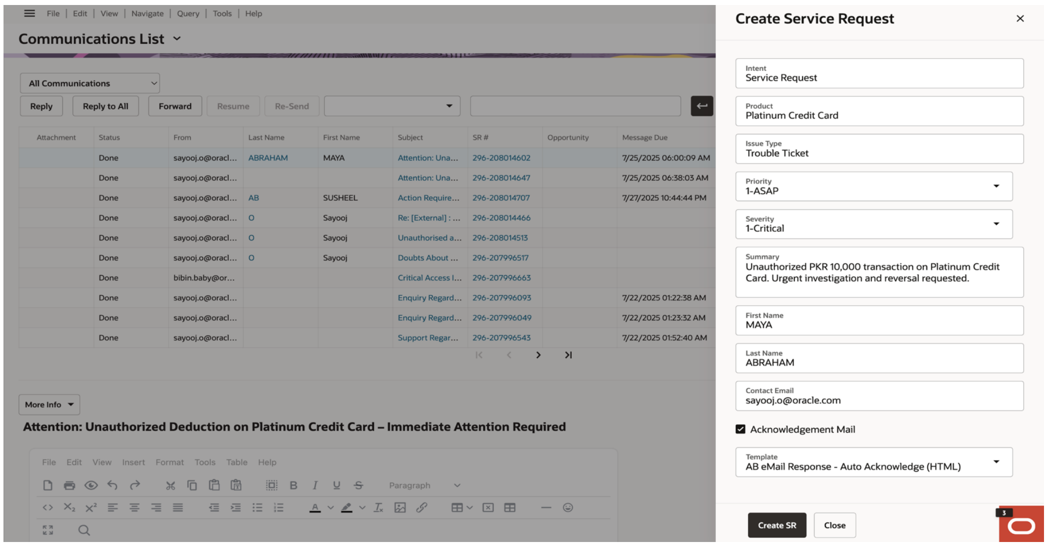
Solution Overview

With the new GenAI integration, working in Siebel Email Response feels smarter, faster, and more intuitive—like having an expert assistant by your side, guiding you through every customer interaction.

Here's what a typical GenAI-powered workflow looks like:

1. **Instant Smart Summaries:** As soon as a customer email or complaint arrives in Siebel, GenAI gets to work. In moments, the email is automatically summarized—long, complex messages are distilled into bullet points or brief highlights. Agents no longer need to scroll through multiple paragraphs hunting for key facts or main issues; everything they need to understand the customer's problem is presented up front.
2. **Critical Details Highlighted Effortlessly:** GenAI doesn't just summarize—it identifies urgency and important data such as product names, account numbers, or time-sensitive requests. This means agents can quickly sense the priority of each interaction, ensuring that the most urgent complaints get the fastest attention.

Figure 1. Generate Email insights flow



Attachment	Status	From	Last Name	First Name	Subject	SR #	Opportunity	Message Due
	Done	sayooj.o@oracl...	ABRAHAM	MAYA	Attention: Una...	296-208014602		7/25/2025 06:00:09 AM
	Done	sayooj.o@oracl...			Attention: Una...	296-208014647		7/25/2025 06:38:03 AM
	Done	sayooj.o@oracl...	AB	SUSHEEL	Action Require...	296-208014707		7/27/2025 10:44:44 PM
	Done	sayooj.o@oracl...	O	Sayooj	Re: [External] : ...	296-208014466		
	Done	sayooj.o@oracl...	O	Sayooj	Unauthorised a...	296-208014513		
	Done	sayooj.o@oracl...	O	Sayooj	Doubts About ...	296-207996517		
	Done	bibin.baby@or...			Critical Access I...	296-207996663		
	Done	sayooj.o@oracl...			Enquiry Regard...	296-207996093		7/22/2025 01:22:38 AM
	Done	sayooj.o@oracl...			Enquiry Regard...	296-207996049		7/22/2025 01:23:32 AM
	Done	sayooj.o@oracl...			Support Regar...	296-207996543		7/22/2025 01:52:40 AM

Attention: Unauthorized Deduction on Platinum Credit Card – Immediate Attention Required

Create Service Request

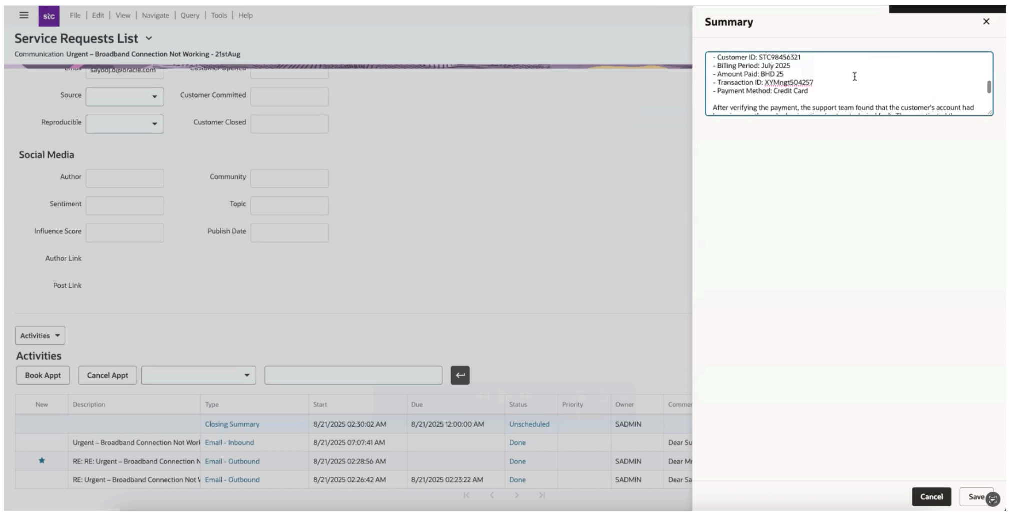
Intent	Service Request
Product	Platinum Credit Card
Issue Type	Trouble Ticket
Priority	1-ASAP
Severity	1-Critical
Summary	Unauthorized PKR 10,000 transaction on Platinum Credit Card. Urgent investigation and reversal requested.
First Name	MAYA
Last Name	ABRAHAM
Contact Email	sayooj.o@oracle.com

[✓] Acknowledgement Mail

Template: AB eMail Response - Auto Acknowledge (HTML)

3. **One-Click Draft Response Generation:** With one click, the agent asks GenAI to draft a suggested reply. GenAI creates a polished, empathetic, and on-topic first draft—often including helpful suggestions, apologies, or next-step instructions as appropriate.
4. **Personalization & Human Touch:** The agent reviews the suggested draft, quickly making any adjustments or adding personal touches. Since the "heavy lifting" is already done by GenAI, agents can focus on tailoring their response, connecting authentically with the customer, and making sure every answer is accurate and on-brand.
5. **Automated Closure Summary Generation:** Once a Service Request (SR) is closed, the agent can generate a precise closure summary based on all SR activities. This summary is then used to draft a transparent, professional closing email to the customer, ensuring clear communication and finality.

Figure 2. Closure Summary Flow



Service Requests List

Communication **Urgent – Broadband Connection Not Working - 21stAug**

Email sayooj.o@oracle.com

Source		Customer Committed	
Reproducible		Customer Closed	

Social Media

Author		Community	
Sentiment		Topic	
Influence Score		Publish Date	
Author Link			
Post Link			

Activities

New	Description	Type	Start	Due	Status	Priority	Owner	Commer
		Closing Summary	8/21/2025 02:30:02 AM	8/21/2025 12:00:00 AM	Unscheduled		SADMIN	
	Urgent – Broadband Connection Not Work	Email - Inbound	8/21/2025 07:07:41 AM		Done			Dear Su
★	RE: RE: Urgent – Broadband Connection N	Email - Outbound	8/21/2025 02:28:56 AM		Done		SADMIN	Dear Mr
	RE: Urgent – Broadband Connection Not V	Email - Outbound	8/21/2025 02:26:42 AM	8/21/2025 02:23:22 AM	Done		SADMIN	Dear Sa

Summary

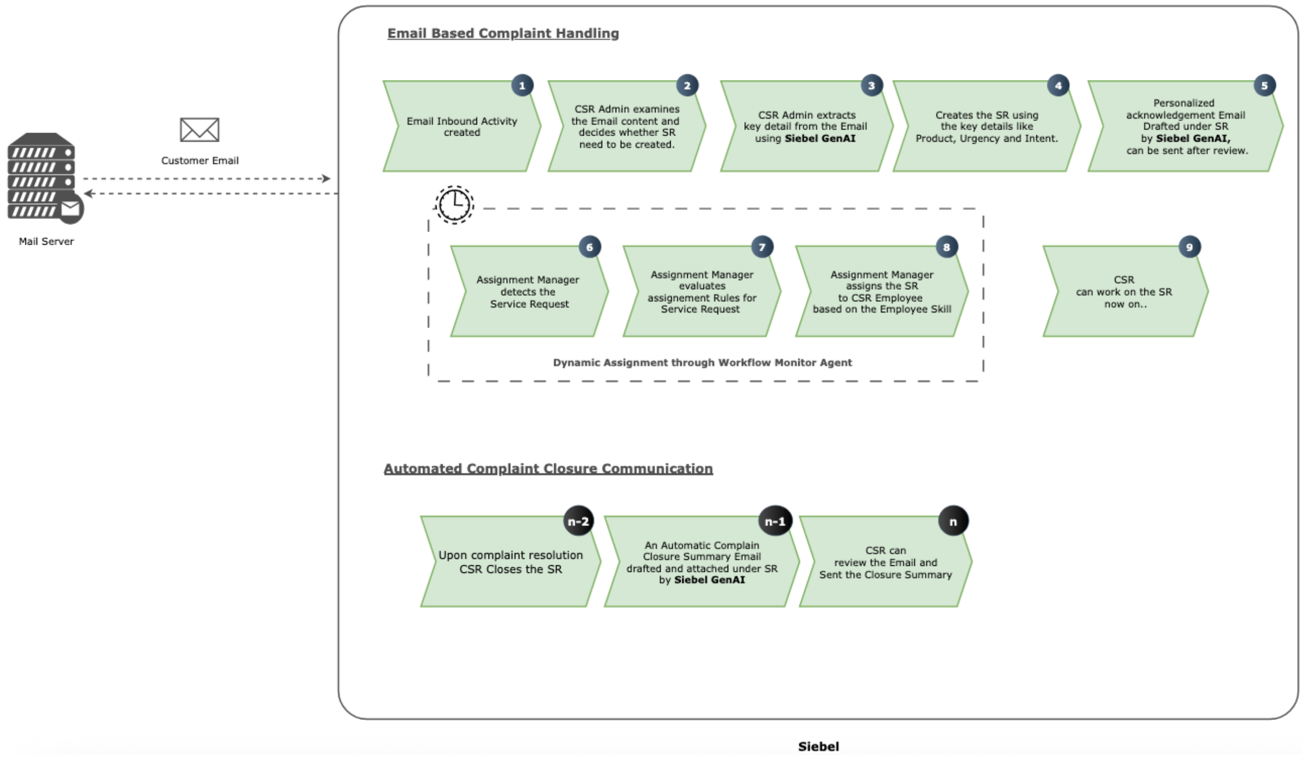
- Customer ID: STC98456321
- Billing Period: July 2025
- Amount Paid: BHD 25
- Transaction ID: XYMngt504257
- Payment Method: Credit Card

After verifying the payment, the support team found that the customer's account had

6. **Seamless Integration, No Added Complexity:** All of this happens inside the familiar Siebel CRM interface—no switching between screens, no copy-paste, no waiting for external tools. The GenAI Adapter runs natively, so agents experience speed, security, and reliability backed by Oracle Cloud.
7. **Learning and Consistency:** Over time, teams notice greater consistency in communication style, fewer errors, and rising customer satisfaction—no matter who's on shift. GenAI suggests closing loops, answering specific questions, or providing links to relevant knowledge articles, so nothing gets missed.

Use Case Deep Dive

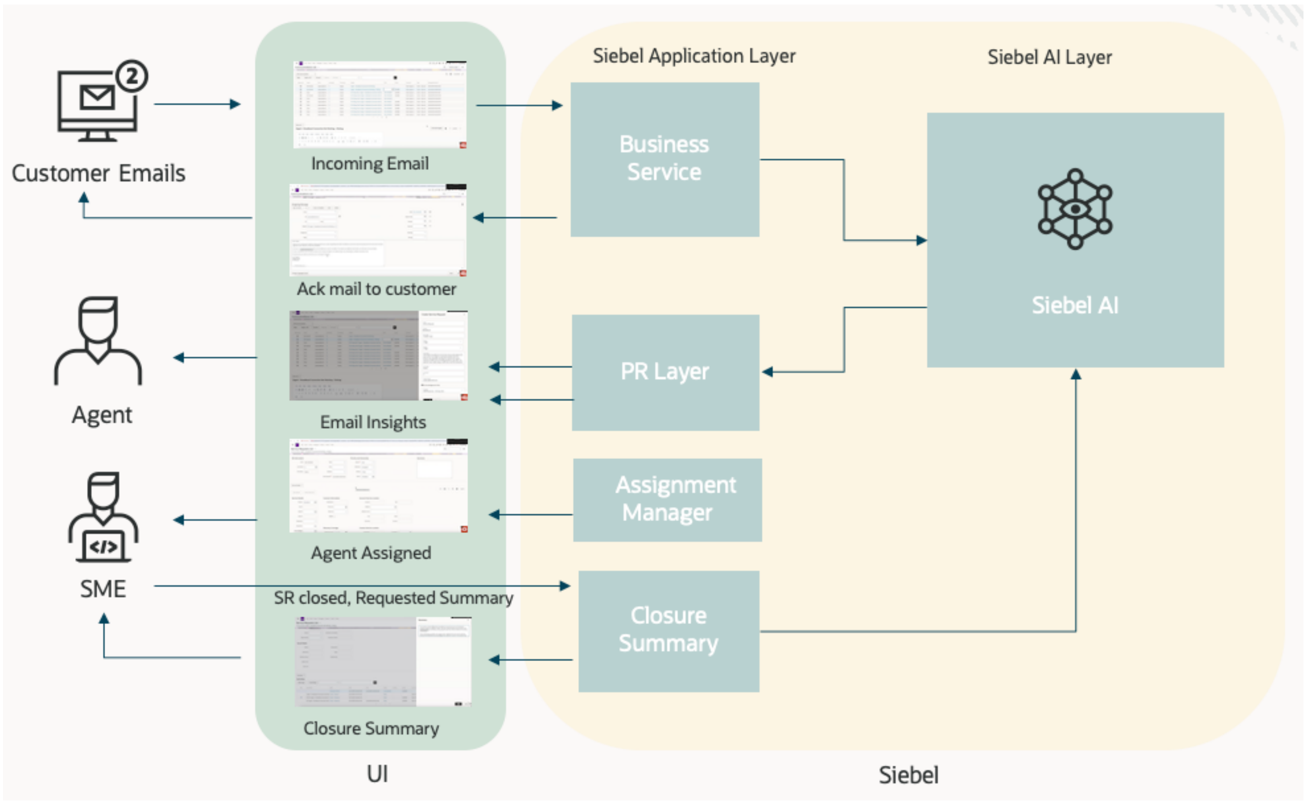
Figure 3. Sample demo flow


Customer Email
Mail Server
Email Based Complaint Handling
1
Email Inbound Activity created
2
CSR Admin examines the Email content and decides whether SR need to be created.
3
CSR Admin extracts key detail from the Email using Siebel GenAI
4
Creates the SR using the key details like Product, Urgency and Intent.
5
Personalized acknowledgement Email Drafted under SR by Siebel GenAI, can be sent after review.
6
Assignment Manager detects the Service Request
7
Assignment Manager evaluates assignement Rules for Service Request
8
Assignment Manager assigns the SR to CSR Employee based on the Employee Skill
9
CSR can work on the SR now on..
Dynamic Assignment through Workflow Monitor Agent
Automated Complaint Closure Communication
n-2
Upon complaint resolution CSR Closes the SR
n-1
An Automatic Complain Closure Summary Email drafted and attached under SR by Siebel GenAI
n
CSR can review the Email and Sent the Closure Summary
Siebel

1. **Automated Email Monitoring** – Siebel is configured to automatically monitor a dedicated support email inbox. Whenever a new email arrives, Siebel extracts the relevant content and syncs it into the system in real-time.
2. **Email Inbound Activity Creation** – Each incoming email is logged as an Inbound Activity within Siebel, providing a structured view for Customer Service Representatives (CSR Admins) to manage.
3. **CSR Review & Decision Point** – The CSR Admin reviews the email content and decides whether a Service Request (SR) needs to be created based on the context and nature of the email.
4. **AI-Assisted Key Information Extraction** – Using Siebel AI capabilities, the CSR extracts key details from the email, such as Product type, Urgency, and Customer Intent, ensuring that all necessary information is captured efficiently.
5. **Service Request Creation** – With the extracted information, the CSR proceeds to create a Service Request (SR), mapping the details into Siebel's SR fields for further processing.
6. **AI-Generated Acknowledgement Draft** – As part of the SR creation process, Siebel AI automatically drafts a personalised acknowledgement email. This draft is reviewed and finalized by the CSR before sending it back to the customer, ensuring timely and professional communication.
7. **Dynamic SR Assignment via Workflow Monitor Agent** – Once the SR is created, Siebel's Workflow Monitor Agent evaluates predefined assignment rules and dynamically assigns the SR to the appropriate Support Agent based on product expertise and workload balancing.
8. **Support Agent Action & Response** – Finally, the assigned Support Agent receives the SR, continues working on the resolution, and can reply directly to the original email thread within Siebel, maintaining a seamless communication loop.

Technical Diagram

Figure 4. Architecture Diagram



Technical Deep Dive: Automating SRs with Siebel AI

This section details the technical architecture and component interactions that enable the intelligent automation of Service Requests (SRs) within the Siebel platform. The core of the system is the Siebel AI module, which integrates with the traditional Siebel UI and backend services to process, analyze, and automate responses to customer interactions.

Architecture Overview

The architecture is composed of three primary layers: the UI (User Interface) Layer, the Siebel Core Services Layer, and the Siebel AI Engine.

- **UI Layer**: This layer represents the user and system touchpoints. It captures incoming customer communications and displays the results of the automated processes to different user roles (Agent and Subject Matter Expert, or SME).

- **Siebel Core Services Layer**: This layer comprises the foundational Siebel business logic and integration components, including Business Services, the PR Layer (Presentation Renderer/Presenter), the Assignment Manager, and the Closure Summary component. These components handle data manipulation, process orchestration, and routing.

- **Siebel AI Engine**: The central intelligence unit responsible for natural language processing (NLP), machine learning (ML) tasks, and generating actionable insights, summaries, and automated responses. This framework is designed for leveraging OCI AI, third-party AI, and Generative AI Services.

Technical Significance
The integration pattern highlights the use of traditional Siebel business components (Business Services, Assignment Manager) as API gateways between the legacy application logic and the modern, black-box AI engine. This decoupling allows the Siebel AI to be a flexible, evolving entity (perhaps a microservice or cloud function) without requiring a deep re-architecture of the core Siebel CRM platform. The reliance on the PR Layer for delivering real-time insights demonstrates a focus on Agent Augmentation, using AI to support the human operator rather than replacing them entirely.

Prerequisites

Please refer appendix I form more details

Implementation

Please refer appendix II form more details

Benefits

The Siebel GenAI integration is designed to move beyond simple automation, delivering tangible improvements in agent efficiency, customer experience, and operational consistency.

- Boost Agent Productivity by Over 30%
  - **Faster Response Time:** Agents save significant time by skipping the manual steps of reading long emails and searching for key data. Instant Smart Summaries and Critical Detail Highlighting deliver the core issue upfront.
  - **Reduced Manual Effort:** The One-Click Draft Response Generation automates the composition of the first draft, allowing agents to focus their energy on review, personalization, and complex problem-solving, not typing out routine replies.
  - **Seamless, Native Workflow:** Because the GenAI Adapter runs natively within Siebel, agents avoid the time-consuming context switching, copy-pasting, and delays associated with external tools.
- Drive Consistent Quality and Customer Satisfaction
  - **High-Quality Communication:** GenAI consistently drafts polished, empathetic, and on-topic responses, ensuring every customer interaction is professional and on-brand, regardless of the agent's experience level.
  - **Transparency and Trust:** The Automated Closure Summary Generation allows agents to quickly provide customers with a clear, accurate account of the resolution, building trust and reducing follow-up queries.
  - **Zero Missed Steps:** The AI helps establish Learning and Consistency by suggesting specific answers, closing loops, or providing links to relevant knowledge articles, minimizing errors and oversight.
- Ensure Security and Reliability
  - **Enterprise-Grade Security:** The adapter's native integration ensures that all data processing occurs securely within the Siebel CRM interface and is backed by the reliability and security of Oracle Cloud.

- **Focus on Personalization:** By handling the "heavy lifting," GenAI enables the Personalization & Human Touch. Agents can dedicate their time to tailoring the final response, fostering an authentic connection, and ensuring absolute accuracy.

Conclusion

The reality of high-volume customer service is that agents often spend more time on low-value, repetitive tasks than on solving complex problems and building customer relationships. The Siebel GenAI integration finally changes this dynamic.

By embedding AI-powered summarization, drafting, and closure generation directly into the familiar Siebel workspace, we're not just offering a new tool—we're delivering a fundamental shift in efficiency. Your team can move beyond simply reacting to emails and instead focus on providing the personalized, high-quality interactions that truly boost satisfaction. This integration is designed to make your agents faster, your service more consistent, and your customers happier, all while leveraging the security and reliability of the Oracle Cloud.

For architects and developers, Appendix A – Technical Implementation Guide provides the detailed architecture, code snippets, and configuration steps referenced throughout this business overview.

Call To Action

Ready to Achieve a 30% Boost in Agent Productivity?

Don't let email volume overwhelm your service team any longer. Take the definitive step toward smarter, AI-powered service today.

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

Appendix I: Prerequisites: Before Implementation

Before beginning the Siebel GenAI Email Response integration, ensure that the following foundational requirements and configurations are fully met and enabled in your environment. Meeting these criteria is essential for successful deployment and proper functioning of the AI features.

Required Configurations Checklist

1. Assignment Manager Setup
    - Verify that Assignment Manager is configured and running correctly to ensure Service Requests are routed efficiently.
2. Email Setup
    - Confirm that inbound and outbound email accounts (SMTP/POP3/IMAP) are fully configured and tested within Siebel for reliable communication.
3. Required Component Group Activation
    - Ensure all necessary Siebel Component Groups (including those supporting Communication Management and the AI Adapter) are enabled and running.
4. Siebel Email Response Configuration
    - Verify that the core Siebel Email Response functionality, including templates and basic response generation, is operational.
5. Siebel AI Services Activation
    - Confirm that the necessary licenses are in place and the external Siebel AI Services (e.g., connectivity to the Oracle Cloud GenAI endpoint) are configured and accessible.
6. Configuring Open UI
    - Ensure the Open UI environment is correctly set up for Presentation Model (PM) and Presentation Renderer (PR) modifications, which are required to display the new AI-powered insights.

Table 1. Prerequisites

Category	Prerequisite/Requirement	Details/Notes
Siebel Environment	Version Min 25.3	Min version for using Siebel GenAI and Open UI services.
	Siebel CRM environment (Server, Gateway, AI, DB) installed and configured	All core Siebel components operational
	Communications Server components enabled (CommInboundRcvr, CommInboundProcessor, CommOutboundMgr)	For email monitoring, processing, and outbound mail
	Siebel Assignment Manager installed, enabled, and configured	For automated, skills-based assignment of Service Requests, activities, emails, etc.
	Siebel administrator/system access	Access to server config, drivers, workflows. This includes the SSH access to VM/Machine where Siebel is installed.

	Network connectivity between Siebel and mail servers (IMAP/POP3/SMTP)	No firewalls/proxies blocking required ports
	Admin access to Siebel environment	Siebel and OS admin rights for configuration and testing
	Network connectivity to email servers (IMAP/POP3/SMTP ports open)	Required: 993 (IMAP), 995 (POP3), 587 (SMTP); no firewall/proxy blocks
Office365/Email Server	Office365/Azure AD tenant or test email infrastructure setup	Test tenant/mailboxes for POC
	Dedicated test mailbox	For inbound/outbound validation
	Azure AD App registration (for OAuth2)	Client ID, Client Secret, Tenant ID, Permissions
	Office365 endpoints accessible: IMAP (993), SMTP (587), POP3 (995)	Network/firewall configuration
	Email server public root certificate (for SSL/TLS)	For use in truststore
OCI Account & Tenancy	Valid Oracle Cloud account and active tenancy	Sign up or use existing OCI tenancy
	Generative AI Supported OCI region	Check that your selected region offers GenAI services (see OCI docs)
OCI Account & Tenancy	**User Access/IAM:** OCI user with required permissions	Appropriate IAM policies that permit use of GenAI resources/compartments
	Compartment where GenAI services will be created/accessed	GenAI resources are created and managed within a compartment
	Sufficient quotas for Generative AI services and relevant resource types	May require a limit increase request via OCI Console
	Network access to OCI endpoints	If accessed from on-prem or Siebel, ensure outbound access to OCI APIs

Appendix II: Technical Implementation Guide

This section outlines the specific configuration and code changes required within the Siebel environment to enable the GenAI-powered email response workflow.

I. GenAI Email Insights Implementation (The Core Functionality)

1. User Experience (UX) and Presentation Renderer (PR) Changes
   - Details on the new applet controls, button configuration, and modifications to the Presentation Model (PM) and Presentation Renderer (PR) files necessary to display the AI-generated summaries and insights within the Siebel UI.
2. Business Service Integration and Data Flow (Backend Logic)
   - Technical specification of the new or modified Business Service responsible for packaging the customer data, invoking the external GenAI API, and processing the returned response properties.

II. Automated Communication and Governance

3. Modifications to the Acknowledgement Mail Process
   - Description of how the GenAI output (e.g., detected urgency or core issue keywords) is used to enhance or select templates for the initial customer acknowledgement mail.
4. Utilizing AI-Generated Summary in the Closure Email
   - Mechanism for calling the GenAI service to generate a final closure summary based on Service Request (SR) activities and how this summary is integrated into the final customer email for maximum transparency.
5. Traceability and Audit Logging of GenAI Activities
   - Outline of the logging and storage strategy for all GenAI-related activities, including storing AI-generated drafts and summaries within the Siebel Service Request record for auditing and governance purposes.

GenAI Email Insights Implementation – UX and PR changes

Create a new Control and add it to the active webtemplate.

- Open Siebel Web Tools in your browser and login.
- Select Workspace or create a new one.
- Navigate to the Applet:
  - Go to the Applet object.
  - Search for and open the target applet.
- Add a New Control: Scroll to the Controls section and click + to add.
  - Name: e.g.,EmailInsights
  - Caption: Get Email Insights
  - HTML Type: MiniButton
- Update Web Template:
  - Open the applet layout and drag the control to the desired location on the form or list layout.
- Validate and Deliver your changes.

The PR File should help in the following ways:

- Button Styling update before Page load and adding event listener.
- When clicked on **Get Insights** Button Retrieve values from the **active applet** (e.g., email body, Sender info).
- Use **OCI GenAI** to extract or infer values such as **Severity**, **Priority**, and **Issue Summary**.
- Pass these insights to create a new **Service Request (SR)** by calling Business Service.
- Finally, **refresh the applet** to display the generated **SR number** for the Email issue.

Prompt to Extract FCL values from Email body:

Gen AI Prompt

```
`You are an intelligent assistant trained to extract structured data from customer service email body. Given an email body, identify and extract the following fields:\n1 **Product** (Fixed Values: 'Personal Loan', 'Platinum Credit Card', 'Child Savings Account', 'Life Insurance')\n2 **SR Type** (Fixed Values: 'Trouble Ticket', 'Technical Problem','Request For Change', 'Request for Information', 'Billing Problem')\n3 **Priority** (Fixed Values: '1-ASAP', '2-High', '3-Medium', '4-Low'). Need to be resolved from urgency in the language used in email body.\n4 **Severity** (Need to be resolved from the severity of the issue. (Fixed Values: '1-Critical', '2-High','3-Medium', '4-Low', '5-Question')) **\n5 **Summary** (Summary of the issue. IMPORTANT: Should not be greater than 100 characters.)\nHere is the email:\n${mailbody}\nReturn your output as a 100% JSON object like this:{product:, sr_type:, priority: ,summary: ,severity:} following all fixed value rules and 100 characters limit for Summary field`
```

Add entry in Manifest Administration and Files:

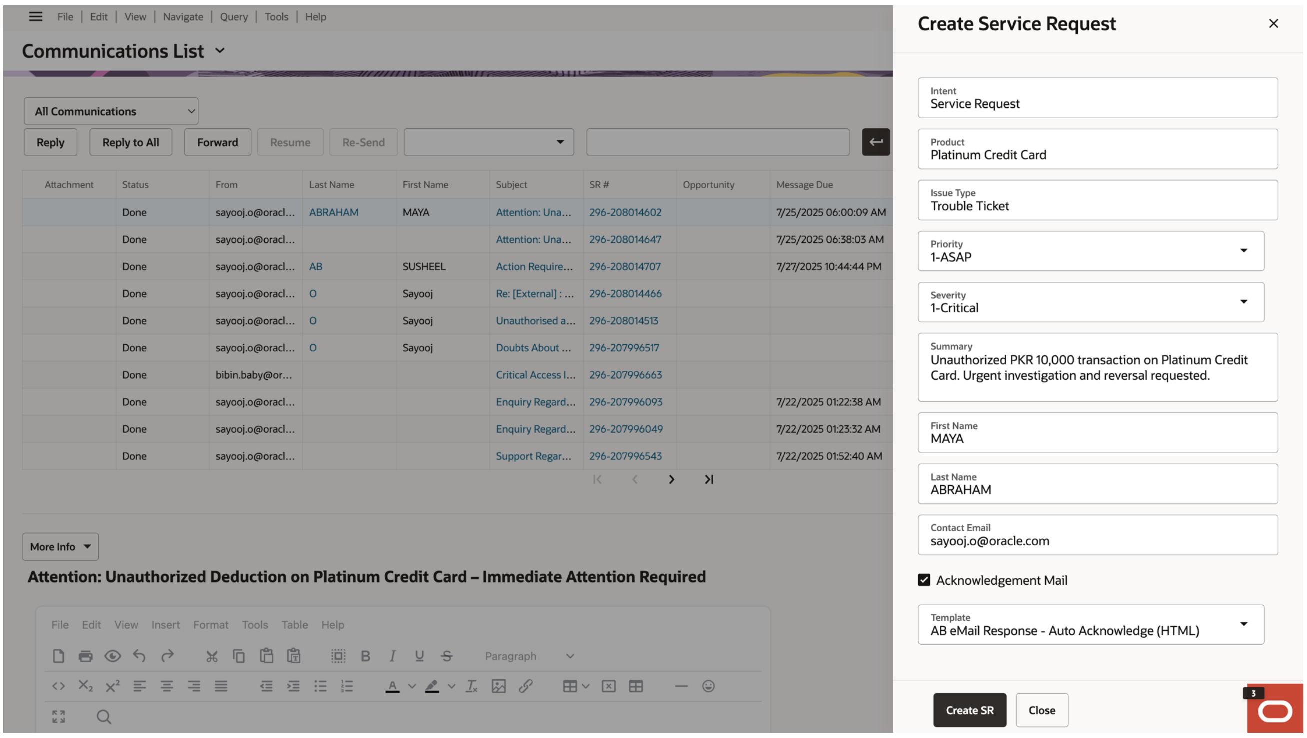
- Type: Applet
- Usage Type: Physical Renderer
- Name: Comm Inbound Item Body Applet
- File: siebel/buttonStylePR.js
- PR File: add the PR File to ses/applicationcontainer_external/siebelwebroot/scripts/siebel

```
if (typeof SiebelAppFacade.buttonStylePR === "undefined") {
    SiebelJS.Namespace('SiebelAppFacade.buttonStylePR');

    define("siebel/buttonStylePR", ["siebel/phyrenderer"], function (PhyRenderer) {
        SiebelAppFacade.buttonStylePR = (function () {

            function buttonStylePR(pm) {
                console.log('buttonStylePR constructor called.');
                SiebelAppFacade.buttonStylePR.superclass.constructor.call(this, pm);
            }

            SiebelJS.Extend(buttonStylePR, SiebelAppFacade.PhysicalRenderer);

            buttonStylePR.prototype.Init = function () {
                console.log('buttonStylePR Init method called.');
                SiebelAppFacade.buttonStylePR.superclass.Init.call(this);
            };

            buttonStylePR.prototype.BindEvents = function () {
                console.log('buttonStylePR BindEvents method called.');
                SiebelAppFacade.buttonStylePR.superclass.BindEvents.call(this);

                var buttonControl = $('#s_1_1_8_0_mb');

                if (buttonControl.length) {
    console.log("Button control found, changing styles");
                var view = SiebelApp.S_App.GetActiveView();
                var appletMap = view.GetAppletMap();
                var applet = appletMap["Comm Inbound Item List Applet"];
                if (applet) {
                    var bc = applet.GetBusComp();
                    var ctrlMap = applet.GetControls();
                    var selectedRecord = {};
                    for (var ctrlName in ctrlMap) {
                        if (!ctrlMap.hasOwnProperty(ctrlName)) continue;
                        var ctrl = ctrlMap[ctrlName];
                        var fieldName = ctrl.GetFieldName && ctrl.GetFieldName();
                        if (fieldName) {
                            try {
                                selectedRecord[fieldName] = bc.GetFieldValue(fieldName);
                            } catch (e) {
                                selectedRecord[fieldName] = "(error)";
                            }
                        }
                    }
                    const isValid =
                        selectedRecord["Type"] === "Email - Inbound" &&
                        selectedRecord["Status"] === "Not Started" &&
                        (!selectedRecord["SR Number"] || selectedRecord["SR Number"].trim() ===
"");
                    if (!isValid) {
                        const emailInsightsBtn = document.getElementById("s_1_1_8_0_mb");
                        if (emailInsightsBtn) {
                            emailInsightsBtn.style.display = "none";
                        }
                    }
                    else {
                        const emailInsightsBtn = document.getElementById("s_1_1_8_0_mb");
                        if (emailInsightsBtn) {
                            emailInsightsBtn.style.display = "inline-block";
                        }
                    }

                }



    var btn = buttonControl[0];

    // Enforce base styling
    btn.style.setProperty("border", "1px solid #888", "important");
    btn.style.setProperty("border-radius", "4px", "important");
    btn.style.setProperty("padding", "6px 12px", "important");
    btn.style.setProperty("font-size", "14px", "important");
    btn.style.setProperty("background-color", "transparent", "important");

    btn.style.setProperty("color", "black", "important");
    btn.style.setProperty("cursor", "pointer", "important");
    btn.style.setProperty("text-decoration", "none", "important");

    // Inject overlay hover effect immediately
    btn.addEventListener("mouseenter", function () {
        btn.style.setProperty("background-color", "rgba(0, 0, 0, 0.05)", "important");
    });

    btn.addEventListener("mouseleave", function () {
        btn.style.setProperty("background-color", "transparent", "important");
    });

    // Attach click listener
    buttonControl.off("click").on("click", function () {
        console.log("Button clicked");
            try {
                    const divId = "CommunicationPanelContainer";
                    const targetDiv = document.getElementById(divId);

                    const existingDrawer = targetDiv.querySelector("oj-c-drawer-popup");
                    if (existingDrawer) targetDiv.removeChild(existingDrawer);

                    const drawer = document.createElement("oj-c-drawer-popup");
                    drawer.setAttribute("edge", "end");
                    drawer.setAttribute("opened", "false");
                    drawer.setAttribute("aria-labelledby", "drawerTitle");
                    drawer.style.display = "flex";
                    drawer.style.flexDirection = "column";
                    drawer.style.width = "650px";

                    const header = document.createElement("div");
                    header.style.display = "flex";
                    header.style.justifyContent = "space-between";
                    header.style.alignItems = "center";
                    header.style.padding = "16px";
                    header.style.borderBottom = "1px solid #ddd";

                    const title = document.createElement("h5");
                    title.id = "drawerTitle";
                    title.textContent = "Create Service Request";
                    title.style.margin = "0";
                    title.style.marginLeft = "16px";
                    title.style.fontWeight = "600";

                    const closeBtn = document.createElement("oj-c-button");
                    closeBtn.setAttribute("display", "icons");
                    closeBtn.setAttribute("chroming", "borderless");
                    closeBtn.setAttribute("label", "Close");
                    const icon = document.createElement("span");
                    icon.setAttribute("slot", "startIcon");
                    icon.className = "oj-ux-ico-close";
                    closeBtn.appendChild(icon);
                    closeBtn.addEventListener("ojAction", () => {
                        drawer.setAttribute("opened", "false");
                    });

                    header.appendChild(title);
                    header.appendChild(closeBtn);

                    const content = document.createElement("div");
                    content.style.padding = "32px";
                    content.style.display = "flex";
                    content.style.justifyContent = "center";
                    content.style.alignItems = "center";
                    content.style.minHeight = "680px";
                    content.style.width = "530px";
                    content.style.flexDirection = "column"; // stack items vertically

                    content.style.minWidth = "350px";
                    const loader = document.createElement("oj-progress-circle");
                    loader.setAttribute("size", "md");
                    loader.setAttribute("value", "-1");
                    content.appendChild(loader);

                    const loadingText = document.createElement("div");
                    loadingText.textContent = "Fetching insights..";
                    loadingText.style.marginTop = "16px";
                    loadingText.style.fontSize = "14px";
                    loadingText.style.color = "#666";
                    content.appendChild(loadingText);

                    const footer = document.createElement("div");
                    footer.style.display = "flex";
                    footer.style.justifyContent = "flex-start";
                    footer.style.paddingLeft = "52px";           // add left padding to shift
right

                    footer.style.gap = "12px";
                    //footer.style.padding = "22px";
                    footer.style.paddingTop = "16px";  // or less
                    footer.style.paddingBottom = "16px";
                    footer.style.marginTop = "auto";   // push it to bottom if wrapper is flex-
column

                    footer.style.borderTop = "1px solid #e0e0e0";
```

```
                        const createBtn = document.createElement("oj-c-button");
                        createBtn.setAttribute("label", "Create SR");
                        createBtn.setAttribute("chroming", "callToAction");
                        createBtn.addEventListener("ojAction", () => {
                            var inputPS = theApplication().NewPropertySet();
                            var OutputPS = theApplication().NewPropertySet();

                            // Utility function to get input value by label
                            function getInputValue(labelHint) {
                                const label = Array.from(document.querySelectorAll('label')).find(l
=> l.textContent.trim() === labelHint);
                                if (label) {
                                    const input =
document.getElementById(label.getAttribute('for'));
                                    return input?.value || '';
                                }
                                return '';
                            }

                            // Extract values templateSelect
                            var email = getInputValue("Contact Email");
                            var priority = document.querySelector("#prioritySelect\\|input")?.value
|| '';

                            var severity = document.querySelector("#severitySelect\\|input")?.value
|| '';

                            var product = getInputValue("Product");
                            var summary = getInputValue("Summary");
                            var lname = getInputValue("Last Name");
                            var fname = getInputValue("First Name");
                            var srtype = getInputValue("Issue Type");
                            const checkboxInput = document.querySelector('oj-c-checkbox
input[type="checkbox"]');
                            const isChecked = checkboxInput?.checked ? "Y" : "N";

                            const selectTemp = document.getElementById("templateSelect");

                                const selectedValue = selectTemp.value; // This will be the "value"
from the selected option



                            var templatetext = selectedValue
                            console.log(templatetext)
                            console.log(severity)
                            const view = SiebelApp.S_App.GetActiveView();
                            let appletMap = view.GetAppletMap();
                            let applet = appletMap["Comm Inbound Item List Applet"];
                            let bc = applet.GetBusComp();
                            var actionid = bc.GetFieldValue("Id");

                            // Set values in Property Set
                            inputPS.SetProperty("email", email);
                            inputPS.SetProperty("priority", priority);
                            inputPS.SetProperty("severity", severity);
                            inputPS.SetProperty("product", product);
                            inputPS.SetProperty("abstract", summary);
                            inputPS.SetProperty("lname", lname);
                            inputPS.SetProperty("fname", fname);
                            inputPS.SetProperty("srtype", srtype);
                            inputPS.SetProperty("actionId", actionid);
                            inputPS.SetProperty("templatetext", templatetext);
                                      inputPS.SetProperty("ackmail", isChecked);
                            console.log(actionid)

                            // Call Business Service
                            var bswm2 = theApplication().GetService("AlliedBankBS");
                            OutputPS = bswm2.InvokeMethod("CreateSR", inputPS);
                            var Retcode = OutputPS.GetProperty("SRId");
                            console.log("Executed " + Retcode);
                            var applet123 = SiebelApp.S_App.GetActiveView().GetAppletMap();
                            var appletname123 = Object.keys(applet123)[0];
                            SiebelApp.S_App.GetActiveView().SetActiveAppletByName(appletname123);
                            var actapplet123 = SiebelApp.S_App.GetActiveView().GetActiveApplet();
                            actapplet123.InvokeMethod("RefreshRecord");
                            console.log("Create SR clicked");
                            drawer.setAttribute("opened", "false");
                            const divId = "CommunicationPanelContainer";
                            const targetDiv = document.getElementById(divId);
                            const existingDrawer = targetDiv.querySelector("oj-c-drawer-popup");
                        });
                        footer.appendChild(createBtn);

                        const cancelBtn = document.createElement("oj-c-button");
                        cancelBtn.setAttribute("label", "Close");
                        cancelBtn.setAttribute("chroming", "outlined");
                        cancelBtn.addEventListener("ojAction", () => {
                            drawer.setAttribute("opened", "false");
                        });

                        footer.appendChild(cancelBtn);


                        drawer.appendChild(header);
                        drawer.appendChild(content);
                        drawer.appendChild(footer);
                        targetDiv.appendChild(drawer);

                        // Open drawer after upgrade
                        customElements.whenDefined("oj-c-drawer-popup").then(() => {
                            requestAnimationFrame(() => {
                                drawer.setAttribute("opened", "true");
                            });
                        });

                        const requireFunc = window.require || window.parent.require;
                        requireFunc(
                            [
                                "oj-c/input-text",
                                "oj-c/text-area",
                                "ojs/ojformlayout",
                                "ojs/ojselectsingle",
                                "ojs/ojarraydataprovider",
                                "oj-c/checkbox"
                            ],
                            function (inputText, textArea, formLayout, selectSingle,
ArrayDataProvider, checkbox) {
                                // Run AI logic in a truly async function
                                (async () => {
                                    try {
                                        await new Promise(resolve => setTimeout(resolve, 100)); //
allow loader to paint
                                        const priorities = [
                                            { value: "1-ASAP", label: "1-ASAP" },
                                            { value: "2-High", label: "2-High" },
                                            { value: "3-Medium", label: "3-Medium" },
                                            { value: "4-Low", label: "4-Low" }
                                        ];
                                        const priorityDP = new ArrayDataProvider(priorities, {
keyAttributes: "value" });

                                        const severities = [
                                            { value: "1-Critical", label: "1-Critical" },
                                            { value: "2-High", label: "2-High" },
                                            { value: "3-Medium", label: "3-Medium" },
                                            { value: "4-Low", label: "4-Low" },
                                            { value: "5-Question", label: "5-Question" }
                                        ];
                                        const severityDP = new ArrayDataProvider(severities, {
    keyAttributes: "value" });

                                        const view = SiebelApp.S_App.GetActiveView();
                                        let appletMap = view.GetAppletMap();
                                        let applet = appletMap["Comm Inbound Item List Applet"];
                                        let bc = applet.GetBusComp();
                                        let ctrlMap = applet.GetControls();
                                        let selectedRecord = {};

                                        for (let ctrlName in ctrlMap) {
                                            if (!ctrlMap.hasOwnProperty(ctrlName)) continue;
                                            let ctrl = ctrlMap[ctrlName];
                                            let fieldName = ctrl.GetFieldName &&
    ctrl.GetFieldName();

                                            if (fieldName) {
                                                try {
                                                    selectedRecord[fieldName] =
    bc.GetFieldValue(fieldName);
                                                } catch (e) {
                                                    selectedRecord[fieldName] = "(error)";
                                                }
                                            }
                                        }

                                        let first = selectedRecord["Contact First Name"];
                                        let last = selectedRecord["Contact Last Name"];
                                        let email = selectedRecord["Email Sender Address"];

                                        applet = appletMap["Comm Inbound Item Body Applet"];
                                        bc = applet.GetBusComp();
                                        ctrlMap = applet.GetControls();
                                        selectedRecord = {};

                                        for (let ctrlName in ctrlMap) {
                                            if (!ctrlMap.hasOwnProperty(ctrlName)) continue;
                                            let ctrl = ctrlMap[ctrlName];
                                            let fieldName = ctrl.GetFieldName &&
    ctrl.GetFieldName();

                                            if (fieldName) {
                                                try {
                                                    selectedRecord[fieldName] =
    bc.GetFieldValue(fieldName);
                                                } catch (e) {
                                                    selectedRecord[fieldName] = "(error)";
                                                }
                                            }
                                        }

                                        let mailbody = selectedRecord["Display Email Body"];

                                        let promptbody = `You are an intelligent assistant trained
    to extract structured data from customer service email body. Given an email body, identify and
    extract the following fields:\n1 **Product** (Fixed Values: 'Personal Loan', 'Platinum Credit
    Card', 'Child Savings Account', 'Life Insurance')\n2 **SR Type** (Fixed Values: 'Trouble
    Ticket', 'Technical Problem','Request For Change', 'Request for Information', 'Billing
    Problem')\n3 **Priority** (Fixed Values: '1-ASAP', '2-High', '3-Medium', '4-Low'). Need to be
    resolved from urgency in the language used in email body.\n4 **Severity** (Need to be resolved
    from the severity of the issue. (Fixed Values: '1-Critical', '2-High','3-Medium', '4-Low', '5-
    Question')) **\n5 **Summary** (Summary of the issue. IMPORTANT: Should not be greater than 100
    characters.)\nHere is the email:\n${mailbody}\nReturn your output as a 100% JSON object like
    this:{product:, sr_type:, priority: ,summary: ,severity:} following all fixed value rules and
    100 characters limit for Summary field`;

                                        let inputPS = theApplication().NewPropertySet();
                                        let OutputPS = theApplication().NewPropertySet();
                                        inputPS.SetProperty("Prompt", promptbody);
                                        inputPS.SetProperty("ModelName", "COHERE");
                                        let bswm2 = theApplication().GetService("AlliedBankBS");
                                        let Output = bswm2.InvokeMethod("InvokeAI", inputPS);
                                        let Retcode = Output.GetProperty("Output");
                                        let jsonObject = JSON.parse(Retcode);
                                        jsonObject["first_name"] = first;
                                        jsonObject["last_name"] = last;
                                        jsonObject["email"] = email;

                                        //await new Promise(resolve => setTimeout(resolve, 100)); //
    allow loader to paint
                                        console.log("going to call rest");
                                        const url =
    "https://phoenix269458.appsdev.fusionappsdphx1.oraclevcn.com:16691/siebel/v1.0/data/Comm%20Packa
    ge/Comm%20Package?searchspec=%5BMedia%20Type%5D%3D'Email'";
                                        const username = <username_here>;
                                        const password = <password_here>;

                                        const response = await fetch(url, {
                                            method: "GET",
                                            headers: {
                                                "Authorization": "Basic " + btoa(username + ":" +
    password),

                                                "Content-Type": "application/json",
                                                "Accept": "application/json"
                                            }
                                        });

                                        if (!response.ok) {
                                            throw new Error(`HTTP ${response.status} -
    ${response.statusText}`);
                                        }

                                        const apiData = await response.json();
                                        console.log(apiData)


                                        const items = apiData.items || [];

                                        const templates = items.map(item => ({
                                            value: item["Template Text"],
                                            label: item["Name"]
                                        }));



                                        const templateDP = new ArrayDataProvider(templates, {
    keyAttributes: "value" });
                                        console.log("template is");
                                        console.log(apiData.items[0]["Template Text"]);


                                        content.innerHTML = `
                              <oj-form-layout max-columns="1" label-edge="top" style="max-width:
    520px; width: 100%;">
                                <oj-c-input-text label-hint="Intent" value="Service Request"></oj-c-
    input-text>
                                <oj-c-input-text label-hint="Product"
    value="${jsonObject.product}"></oj-c-input-text>
                                <oj-c-input-text label-hint="Issue Type"
    value="${jsonObject.sr_type}"></oj-c-input-text>
                                <oj-select-single id="prioritySelect" label-hint="Priority"
    value="${jsonObject.priority}" class="oj-form-control-max-width-md"></oj-select-single>
                                <oj-select-single id="severitySelect" label-hint="Severity"
    value="${jsonObject.severity}" class="oj-form-control-max-width-md"></oj-select-single>
    <oj-c-text-area label-hint="Summary" rows="3" value="${jsonObject.summary?.substring(0,
    100)}"></oj-c-text-area>
                                <oj-c-input-text label-hint="First Name"
    value="${jsonObject.first_name}"></oj-c-input-text>
```

```
                            <oj-c-input-text label-hint="Last Name"
value="${jsonObject.last_name}"></oj-c-input-text>
                            <oj-c-input-text label-hint="Contact Email"
value="${jsonObject.email}"></oj-c-input-text>
<oj-c-checkbox label-hint="Acknowledgement Mail" value="true">Acknowledgement Mail</oj-c-
checkbox>
<oj-select-single id="templateSelect" label-hint="Template" value="${apiData.items[0]["Template
Text"]}" class="oj-form-control-max-width-md"></oj-select-single>

                          </oj-form-layout>
                        `;

                                    const selectElem =
document.getElementById("prioritySelect");
                                    if (selectElem) {
                                        selectElem.data = priorityDP;
                                    }
                                    const selectTemp =
document.getElementById("templateSelect");
                                    if (selectTemp) {
                                        selectTemp.data = templateDP;
 selectTemp.value = apiData.items[0]["Template Text"]
                                    }

                                    const selectSev = document.getElementById("severitySelect");
                                    if (selectSev) {
                                        selectSev.data = severityDP;
                                    }

                                } catch (err) {
                                    content.innerHTML = `<div style="color:red;">Error:
${err.message}</div>`;
                                    console.error("AI Drawer Error:", err);
                                }
                            })();
                        }
                    );
                } catch (error) {
                    console.error("Error in button logic:", error.message);
                    alert("An error occurred: " + error.message);
                }
    }.bind(this));
}
 else {
                    console.log("Button control not found.");
                }
            };
            return buttonStylePR;

        }());
        return "SiebelAppFacade.buttonStylePR";
    });
}
```

Figure 5. This is the final UI after PR changes



Attachment	Status	From	Last Name	First Name	Subject	SR #	Opportunity	Message Due
	Done	sayooj.o@oracl...	ABRAHAM	MAYA	Attention: Una...	296-208014602		7/25/2025 06:00:09 AM
	Done	sayooj.o@oracl...			Attention: Una...	296-208014647		7/25/2025 06:38:03 AM
	Done	sayooj.o@oracl...	AB	SUSHEEL	Action Require...	296-208014707		7/27/2025 10:44:44 PM
	Done	sayooj.o@oracl...	O	Sayooj	Re: [External] : ...	296-208014466		
	Done	sayooj.o@oracl...	O	Sayooj	Unauthorised a...	296-208014513		
	Done	sayooj.o@oracl...	O	Sayooj	Doubts About ...	296-207996517		
	Done	bibin.baby@or...			Critical Access I...	296-207996663		
	Done	sayooj.o@oracl...			Enquiry Regard...	296-207996093		7/22/2025 01:22:38 AM
	Done	sayooj.o@oracl...			Enquiry Regard...	296-207996049		7/22/2025 01:23:32 AM
	Done	sayooj.o@oracl...			Support Regar...	296-207996543		7/22/2025 01:52:40 AM

Attention: Unauthorized Deduction on Platinum Credit Card – Immediate Attention Required

Create Service Request

Intent	Service Request
Product	Platinum Credit Card
Issue Type	Trouble Ticket
Priority	1-ASAP
Severity	1-Critical
Summary	Unauthorized PKR 10,000 transaction on Platinum Credit Card. Urgent investigation and reversal requested.
First Name	MAYA
Last Name	ABRAHAM
Contact Email	sayooj.o@oracle.com
Acknowledgement Mail	✓
Template	AB eMail Response - Auto Acknowledge (HTML)

Get Email Insights Button – Business Service

To create a Business Service in Siebel Web Tools, follow these steps:

1. Log in to Siebel Web Tools

2. Navigate to "Business Services"

3. Create a New Business Service

4. Define Methods

- In the **Methods** tab (or view), click **Add.**
- Specify the method **Name** (e.g., "CreateSR").
- You can create multiple methods as needed.

5. Implement Scripts

- Click on the Business Service you created
- Under settings go to server scripts.
- Under Preinvoke_method add below code

```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
   if(MethodName === "InvokeGenericAI"){
        InvokeGenericAI(Inputs, Outputs);
        return (CancelOperation);
```

```
            }if(MethodName === "InvokeAI"){
                InvokeAI(Inputs, Outputs);
                return (CancelOperation);
            } if(MethodName === "extractTextField"){
                extractTextField(Inputs, Outputs);
                return (CancelOperation);
            } if(MethodName === "cleanPrompt"){
                cleanPrompt(Inputs, Outputs);
                return (CancelOperation);
            } if(MethodName === "clip"){
                clip(Inputs, Outputs);
                return (CancelOperation);
            } if(MethodName === "StringTrim"){
                StringTrim(Inputs, Outputs);
                return (CancelOperation);
            }
            if (MethodName == "CreateSR"){
                CreateSR(Inputs, Outputs)
                return (CancelOperation);
            }
            return (ContinueOperation);
        }
```

- Add your script under General → "CreateSR"
-

```
        function CreateSR(Inputs, Outputs) {
                var contactBO = TheApplication().GetBusObject("Contact");
                var contactBC = contactBO.GetBusComp("Contact");
                var contactId;
                var email = Inputs.GetProperty("email");
                var priority = Inputs.GetProperty("priority");
                var product = Inputs.GetProperty("product");
                var abstract1 = Inputs.GetProperty("abstract");
                var srtype = Inputs.GetProperty("srtype");
                var lname1 = Inputs.GetProperty("lname");
                var fname1 = Inputs.GetProperty("fname");
                var emailId = Inputs.GetProperty("actionId");
                var sendackmail = Inputs.GetProperty("ackmail");
                var templatetext = Inputs.GetProperty("templatetext");
                var severity = Inputs.GetProperty("severity");
                var emailSubject;
                var emailBody;
                var emailToLine;
                var creationDate;
                with (contactBC) {
                    ActivateField("Email Address");
                    ActivateField("Id");
                    ClearToQuery();
                    SetSearchSpec("Email Address", email);
                    ExecuteQuery(ForwardOnly);
                    if (FirstRecord()) {
                        contactId = GetFieldValue("Id");
                    }
                    else {
                        // Create new contact
                        NewRecord(NewAfter);
                        contactBC.SetFieldValue("First Name", fname1);
                        contactBC.SetFieldValue("Last Name", lname1);
                        contactBC.SetFieldValue("Email Address", email);
                        WriteRecord();
                        contactId = contactBC.GetFieldValue("Id");
                    }
                }
                var SRBO = TheApplication().GetBusObject("Service Request");
                var SRBC = SRBO.GetBusComp("Service Request");
                SRBC.NewRecord(NewAfter);
                SRBC.SetFieldValue("Status", "Open");
                SRBC.SetFieldValue("Abstract", abstract1);
                SRBC.SetFieldValue("Priority", priority);
                //SRBC.SetFieldValue("Description", desc);
                SRBC.SetFieldValue("Contact Id", contactId);
                SRBC.SetFieldValue("Severity", severity);
                SRBC.SetFieldValue("Owner", "");
                SRBC.SetFieldValue("SR Type", srtype);
                SRBC.SetFieldValue("Product", product);
                SRBC.WriteRecord();
                var srid = SRBC.GetFieldValue("Id")
                var srNo = SRBC.GetFieldValue("SR Number")
                var EmailBO = TheApplication().GetBusObject("Action");
                var EmailBC = EmailBO.GetBusComp("Action");
                with (EmailBC) {
                   ActivateField("Id");
                   ActivateField("SR Number");
                   ActivateField("Status");
                   ActivateField("Comment");
                   ActivateField("Description");
                   ActivateField("Email To Line");
                   ActivateField("Creation Date");
```

```
            ClearToQuery();
            SetViewMode(AllView);
            SetSearchSpec("Id", emailId);
            ExecuteQuery(ForwardOnly);
            if (FirstRecord()) {
               TheApplication().SetProfileAttr("CSR15","SR Number "+srNo+". Email id
"+emailId);
               emailSubject = EmailBC.GetFieldValue("Description")
               emailBody = EmailBC.GetFieldValue("Comment")
               emailToLine = EmailBC.GetFieldValue("Email To Line")
               creationDate = EmailBC.GetFieldValue("Creation Date")
               EmailBC.SetFieldValue("SR Number", srNo);
               EmailBC.SetFieldValue("Status", "Done");
               EmailBC.WriteRecord();
              }
            }
         if (sendackmail == "Y") {
             var ActionBO = TheApplication().GetBusObject("Action");
             var ActionBC = ActionBO.GetBusComp("Action");
             var promptbody = "Generate a professional acknowledgment email using the
following template.Replace the placeholders and HTML tags in the following template with
the provided values.Please ensure the email is grammatically correct and professionally
written. Template : "+ templatetext+" ; Customer Name: "+fname1+" ; Issue summary -
"+abstract1+" ; SR number : "+srNo;
             var inputPS = TheApplication().NewPropertySet();
             var OutputPS = TheApplication().NewPropertySet();
             inputPS.SetProperty("Prompt", promptbody);
             InvokeGenericAI(inputPS, OutputPS);
             var Retcode = OutputPS.GetProperty("Output");
             TheApplication().SetProfileAttr("CSR11",Retcode);
             var ackMailBody = Retcode+"\r\n\r\n\r\n\r\n-----Original Message-----
\r\n\r\nFrom: "+email+"\r\nSent: "+creationDate+"\r\nTo: "+emailToLine+"\r\nSubject:
"+emailSubject+"\r\n\r\n\r\n"+emailBody;
             ActionBC.NewRecord(NewAfter);
             ActionBC.SetFieldValue("Type", "Email - Outbound");
             ActionBC.SetFieldValue("Description", "Re: "+emailSubject);
             ActionBC.SetFieldValue("Display", "Communication and Activities");
             ActionBC.SetFieldValue("Comment", Retcode);
             ActionBC.SetFieldValue("Email Body", ackMailBody);
             ActionBC.SetFieldValue("SR Number", srNo);
             ActionBC.SetFieldValue("Status", "Draft");
             ActionBC.SetFieldValue("Primary Contact Id", contactId);
             ActionBC.SetFieldValue("Email To Line", email);
             ActionBC.WriteRecord();
         }
Outputs.SetProperty("Status", "done");
```

```
        }
```

- Compile/Activate. Save your changes.If your environment uses automatic deployment, your changes take effect immediately. Otherwise, follow your deployment procedures.

6. Test the Business Service
Use the Siebel application UI or Siebel COM/CLI/Script Call to test your business service.

Business service Invocation

To invoke this business service from another script do below changes

1. In Siebel Tools, navigate to the "Application" object.
2. Search for "Siebel Universal Agent"
3. Go to "Application User Prop" section
4. Add new user property. Increment value of ClientBusinessServiceXX and give your Business service name.

Acknowledgement mail

Prompt

```
Generate a professional acknowledgment email using the following template.Replace the
placeholders and HTML tags in the following template with the provided values.Please ensure the
email is grammatically correct and professionally written. Template : "+ templatetext+" ;
Customer Name: "+fname1+" ; Issue summary - "+abstract1
```

Code to generate acknowledgment email is added in the business service.

Closure Email

This feature is available out-of-the-box. Please go through Siebel bookshelf - https://docs.oracle.com/cd/G30556_01/books/AppsAdmin/c-Use-Case-2-Generate-Transfer-and-Closing-Summary.html - to know more about this feature.

Traceability

All sent emails should be logged in the CRM system against the customer's complaint record for full traceability

Siebel Email Response supports saving outbound emails in Siebel application. The email content, including attachments, is associated with a Siebel activity record that tracks the send operation (out-of-the-box)

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