Clinical Protocol Management with Deviation Summarization Recipe

Executive Summary

As clinical trials become more complex, managing protocol deviations efficiently is key to ensuring data integrity and compliance. When deviation records in Siebel Clinical Trial Management Systems (CTMS) are entered in free-text fields, they often become verbose and difficult to manage. To address this, a Generative AI-based solution is proposed that parses and summarizes protocol deviation records. This system presents the summarized information to the user for review and editing, ensuring accuracy and clarity. Once finalized, the summary is automatically applied to the parent protocol record, reducing manual input and improving overall protocol management efficiency. This approach leverages AI to streamline the process and ensure precise documentation.

Introduction

In Clinical Trial Management Systems (CTMS), managing protocol deviations can be a challenging and labor-intensive process. Detailed deviation descriptions and multiple records often result in verbose entries that are difficult to interpret and summarize. Introducing a Generative AI-based solution for protocol deviation summarization significantly streamlines this process, enhancing clarity and efficiency.

For example, when protocol deviation records are entered into the CTMS, the Clinical Research Associate (CRA) may need to go through the entire older deviation records for the same Protocol record which is very time consuming. But using this system the CRA can click on "Generate Summary" button in the protocol deviation records list applet, Gen AI model parses through the detailed descriptions of each Protocol Deviation, identifies key points, and generates a concise summary. This summary is then presented to the user for review and editing, allowing for interactive refinement. Once approved, the final summary is stamped onto the parent protocol record, ensuring seamless integration with minimal manual intervention.

Challenge: Streamlining Protocol Management with Effective Deviation Summarization

Managing protocol deviations effectively poses several challenges, including verbose descriptions, multiple deviation records, and difficulty in maintaining accurate summaries. These inefficiencies can lead to miscommunication, increased manual effort, and compliance risks in clinical trial workflows. A Generative AI-based summarization solution addresses these issues by parsing detailed deviation records, generating concise summaries, and integrating them seamlessly into protocol management systems. This innovative approach reduces manual workload, ensures consistency in documentation, and improves overall trial efficiency.

Workflow Overview

- CRA selects a Protocol record with several Protocol deviation records
- CRA clicks on Generate Summary Button and gets a Popup of AI Generated Summary of all the Protocol deviations with option to Make required changes
- CRA saves the Summary for that particular Protocol deviation record

Key Technologies Involved

item	Description
Siebel AI Framework	Siebel AI framework is enabled in the Siebel CTMS Env which provides Gen AI REST Endpoint
CTMS UI Customization	SIebel Open UI PR file for UI Customizations, Protocol Deviation records collection and Business Service invocation

Implementation Strategy

The implementation of this flow involves

1. Adding a new Text Field Area in Clinical Protocol Business Object - "Generated Summary"
2. PR File for collecting protocol deviation, invoking Siebel AI Framework REST Endpoint and UI Customization

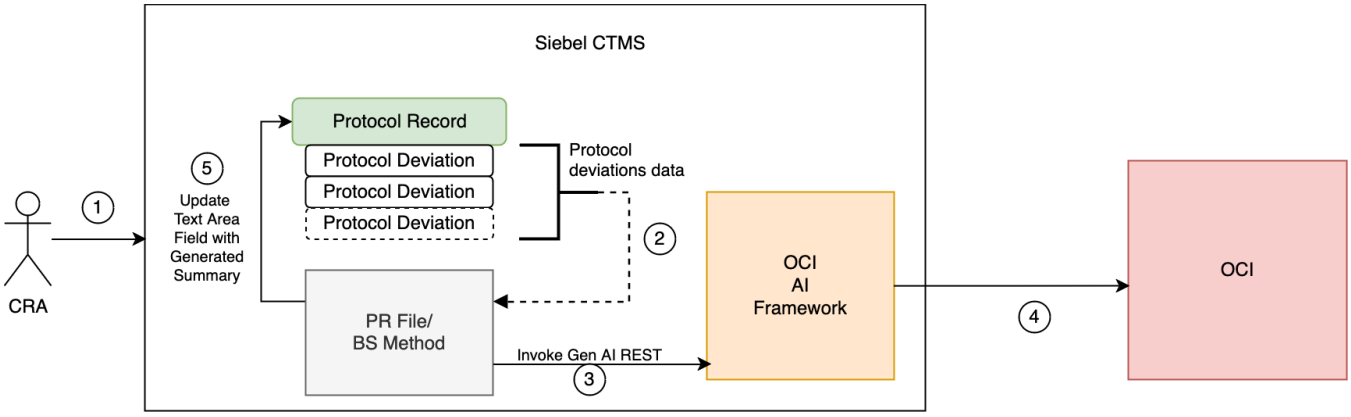
Architecture Diagram

This functionality can be implemented in 2 ways:

1) Leveraging Siebel OCI AI Framework

2) Creating a wrapper Node JS Service that calls the OCI APIs

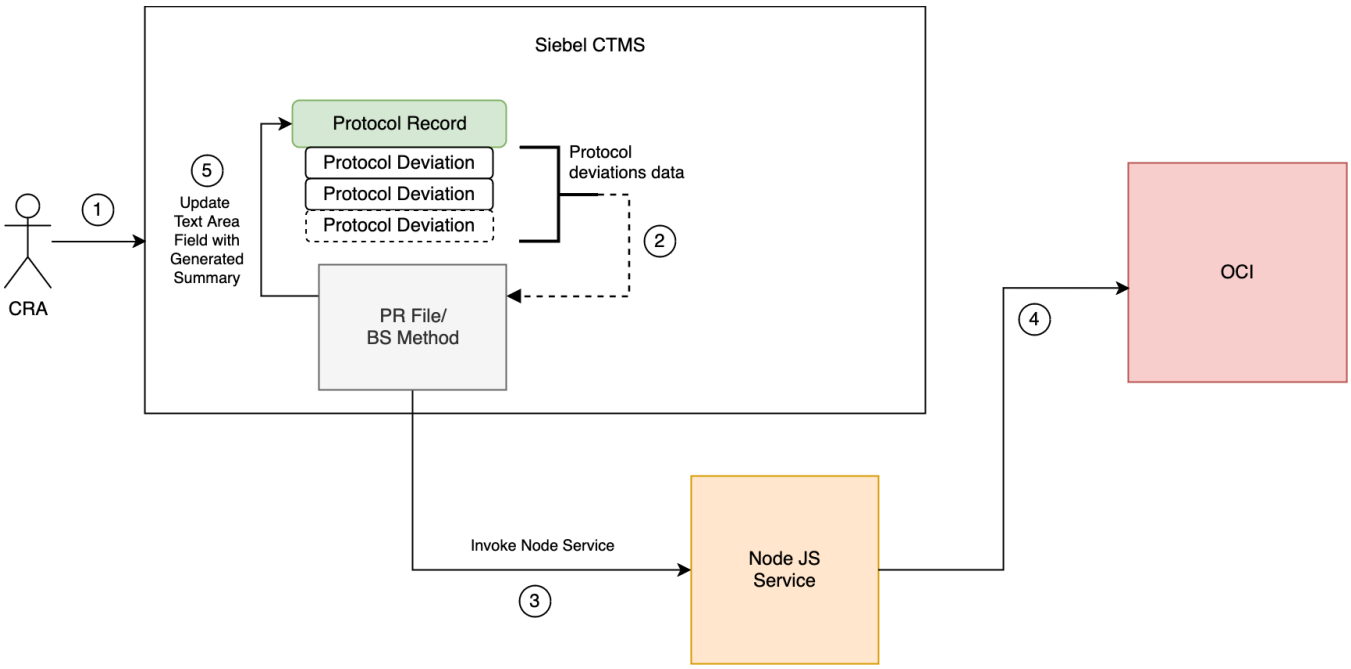
Leveraging Siebel OCI AI Framework (Recommended)



Steps

1. CRA Clicks on Generate Summary Button on the Protocol deviations list page Applet
2. PR File or BS Scripts collects all the Protocol Deviation records description data
3. OCI AI Framework - Gen AI Endpoint is invoked and the Collective Protocol deviation records data is passed
4. OCI AI Framwork calls the OCI Gen AI and gets the summary
5. PR File/ BS Scripts update the Protocol Records Text Area Field with generated Summary

Leveraging OCI Gen AI Service via External Node JS Service

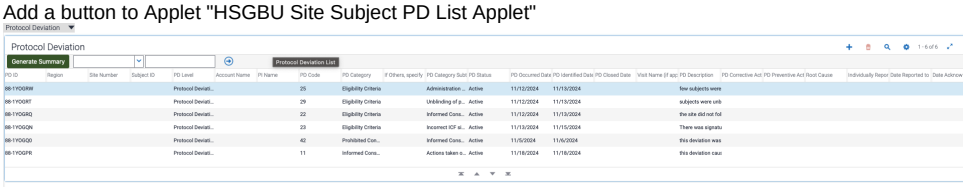


Steps

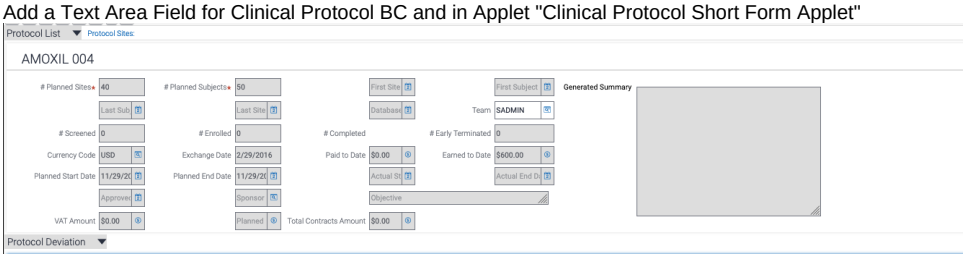
1. CRA Clicks on Generate Summary Button on the Protocol deviations list page Applet
2. PR File or BS Scripts collects all the Protocol Deviation records description data
3. External Node JS service is invoked and the Collective Protocol deviation records data is passed
4. Node JS service calls the OCI Gen AI and gets the summary
5. PR File/ BS Scripts update the Protocol Records Text Area Field with generated Summary

Siebel Customisations

1. Add a button to Applet "HSGBU Site Subject PD List Applet"



2. Add a Text Area Field for Clinical Protocol BC and in Applet "Clinical Protocol Short Form Applet"



3. In order to handle Generate Summary action we can take 2 approaches
    a. Writing Business Service Method that gets invoked on clicking "Generate Summary" Button. (Recommended)
    b. PR File registers event listener to handle Button click and execute the Summary generation and UI Changes.
    Physical Renderer Code:

```
// Ensure SiebelJS Namespace and Definition
if (typeof SiebelAppFacade.ProtocolDeviationDetailAppletCustomPR === "undefined") {
    SiebelJS.Namespace('SiebelAppFacade.ProtocolDeviationDetailAppletCustomPR');

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

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

            SiebelJS.Extend(ProtocolDeviationDetailAppletCustomPR, SiebelAppFacade.
PhysicalRenderer);

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

            // BindEvents Method with jQuery Event Registration
            ProtocolDeviationDetailAppletCustomPR.prototype.BindEvents = function () {
                console.log('ProtocolDeviationDetailAppletCustomPR BindEvents method called.');
                SiebelAppFacade.ProtocolDeviationDetailAppletCustomPR.superclass.BindEvents.call
(this);

                // Button Control (Ensure button ID is accurate for your configuration)
                var buttonControl = $('#s_1_1_3_0_Ctrl'); // Replace with appropriate control ID

                if (buttonControl.length) {
                    console.log("Button control found. Setting up jQuery click event.");

                    buttonControl.off("click").on("click", function () {
                        console.log("Generate button clicked.");
                        try {
                            // Get the active applet and business component
                            const applet = SiebelApp.S_App.GetActiveView().GetActiveApplet();
```

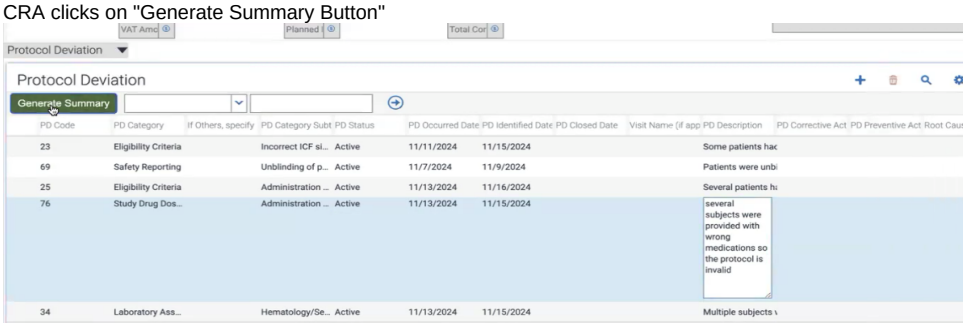
```
                    if (!applet) {
                        console.error("No active applet found.");
                        return;
                    }

                    const busComp = applet.GetBusComp();
                    if (!busComp) {
                        console.error("No business component associated with the applet.");
                        return;
                    }

                    // Perform query to retrieve records
                    busComp.SetSearchSpec("Id", ""); // Adjust search spec if needed
                    const recs = busComp.GetRawRecords();

                    // Format records into a string
                    const appendedString = recs
                        .map((item, index) => `${index + 1}. ${item[16]}`)
                        .join("\n");

                    // Call the popup creation function
                    createPopup(appendedString);
                } catch (error) {
                    console.error("Error during the operation:", error);
                }
            });
        } else {
            console.error("Button control not found.");
        }
    };

    // Function to Create Popup
    async function createPopup(appendedString) {
        // Fetch Chat Response
        const chatResponse = await fetchChatResponse(appendedString);

        // Popup and Overlay
        const overlay = $("<div>").css({
            position: "fixed",
            top: "0",
            left: "0",
            width: "100%",
            height: "100%",
            backgroundColor: "rgba(0, 0, 0, 0.5)",
            zIndex: "999"
        });

        const popup = $("<div>").css({
            position: "fixed",
            top: "50%",
            left: "50%",
            transform: "translate(-50%, -50%)",
            width: "520px",
            padding: "26px",
            boxShadow: "0 6px 12px rgba(0, 0, 0, 0.3)",
            backgroundColor: "#fff",
            border: "1px solid #ccc",
            zIndex: "1000",
            fontFamily: "Arial, sans-serif"
        });

        // Add Content to Popup
        popup.append($("<h4>").text("AI Generated Summary").css({
            marginTop: "0",
            marginBottom: "10px",
            fontSize: "16px",
            textAlign: "left",
            color: "#333"
        }));

        const textarea = $("<textarea>").css({
```

```
            width: "100%",
            height: "195px",
            marginBottom: "10px",
            fontSize: "14px",
            resize: "none",
            border: "1px solid black",
            borderRadius: "0",
            padding: "8px",
            overflow: "auto",
            whiteSpace: "pre-wrap"
        }).val(chatResponse);

        popup.append(textarea);

        // Add Buttons
        const saveButton = $("<button>").text("Save").css({
            marginRight: "10px",
            padding: "8px 10px",
            border: "none",
            backgroundColor: "#385427",
            color: "#fff",
            fontSize: "15px",
            cursor: "pointer",
            borderRadius: "4px"
        }).on("click", function () {
            // Save Logic
            const updatedText = textarea.val();
            console.log("Updated Text:", updatedText);
            // Remove Popup and Overlay
            overlay.remove();
            popup.remove();
        });

        const exitButton = $("<button>").text("Exit").css({
            padding: "8px 10px",
            border: "none",
            backgroundColor: "#c0533f",
            color: "#fff",
            fontSize: "15px",
            cursor: "pointer",
            borderRadius: "4px"
        }).on("click", function () {
            // Close Logic
            overlay.remove();
            popup.remove();
        });

        popup.append(saveButton, exitButton);
        $("body").append(overlay, popup);
    }

    // Fetch Chat Response
    async function fetchChatResponse(inputText) {
        try {
            const response = await fetch("http://127.0.0.1:5000/chat-response", {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify({ input: inputText })
            });

            if (response.ok) {
                const data = await response.json();
                return data.text.replace(/^"(.*)"$/, '$1'); // Clean up quotes
            } else {
                console.error("Failed to fetch chat response:", response.statusText);
                return inputText;
            }
        } catch (error) {
            console.error("Error during POST request:", error);
```

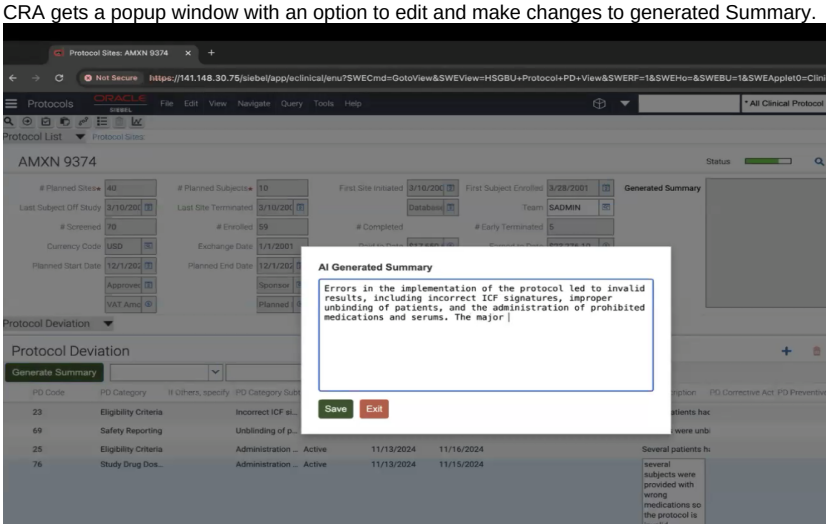
```
                    return inputText;
                }
            }

            return ProtocolDeviationDetailAppletCustomPR;

        }());

        return "SiebelAppFacade.ProtocolDeviationDetailAppletCustomPR";
    });
}
```

Functionality

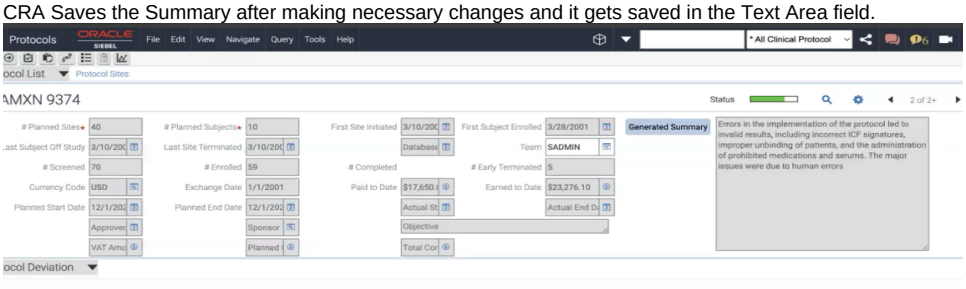
1. CRA clicks on "Generate Summary Button"



2. CRA gets a popup window with an option to edit and make changes to generated Summary.



3. CRA Saves the Summary after making necessary changes and it gets saved in the Text Area field.



Benefits

- **Streamlined Protocol Management**
  By automating the summarization of verbose protocol deviation records, the system reduces manual effort, allowing clinical teams to save time and focus on critical trial activities.
- **Enhanced Data Accuracy**
  The AI-powered summarization ensures that only accurate and relevant details are captured, minimizing errors and inconsistencies across protocol documentation.

- **Improved User Efficiency**
  With an intuitive review and editing interface, users can quickly refine and approve summaries, ensuring a seamless workflow and reducing cognitive load.

Future Possibilities and Innovations

1. Predict Total % Severity as a BO field in cumulative Protocol deviations.

Limitations

1. A limitation of the AI-driven summarization feature is that the generated summary may miss crucial deviations.

Conclusion

By leveraging Generative AI for protocol deviation summarization, CRA can significantly enhance the efficiency and accuracy of clinical trial management. The seamless integration of AI-driven summarization reduces manual effort, ensures consistency in documentation, and improves compliance. This innovative approach streamlines protocol deviation handling, enabling Clinical Research Associates to focus on critical decision-making while maintaining data integrity. Automating summary generation and integration into protocol records not only saves time and costs but also enhances the overall efficiency of clinical trial workflows, ensuring more effective and transparent protocol management.