Integrating Siebel CRM with Microsoft Teams for Real-Time Notifications and Improved Collaboration – Recipe

May, 2025, Version [1.0]

Purpose statement

This document has been created by the Siebel Center of Excellence team for informational purposes only. It aims 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

Summary

This whitepaper outlines the integration of Siebel CRM with Microsoft Teams to enhance collaboration and streamline notification workflows. The integration facilitates seamless real-time communication by enabling MS Teams to notify users of significant changes made within the Siebel CRM system. It leverages RESTful APIs, adaptive cards, and dynamic deep linking, ensuring that all Siebel updates are communicated to relevant stakeholders directly in MS Teams.

Introduction

Siebel CRM is a robust customer relationship management platform that handles a wide variety of business processes. Integrating Siebel with Microsoft Teams enables users to remain updated about critical business events without having to switch between applications.

The purpose of this integration is to enhance productivity, improve communication, and reduce operational inefficiencies by pushing Siebel CRM updates to Microsoft Teams in a structured and actionable format.

Objectives of the Integration

The key objectives of this integration include:

- Real-time notifications to users in MS Teams upon key events within Siebel.
- Ensuring dynamic deep linking to specific records in Siebel from MS Teams, allowing users to access critical data quickly.
- Streamlining communication and collaboration within MS Teams by providing users with updates formatted using adaptive cards.
- Offering the flexibility to notify the correct set of users based on the changes made in Siebel.

Pre-requisites

For Development

- MS Teams Toolkit Extension should be installed in Visual Studio Code
- Microsoft 365 Developer account with Custom app upload permission enabled (For Testing the Integration with Microsoft Teams)
- Azure Entra ID account with access to Azure services (To create the Bot using Microsoft Bot Framework)

For Usage

- The user should be using the same email to login to both Siebel and MS Teams

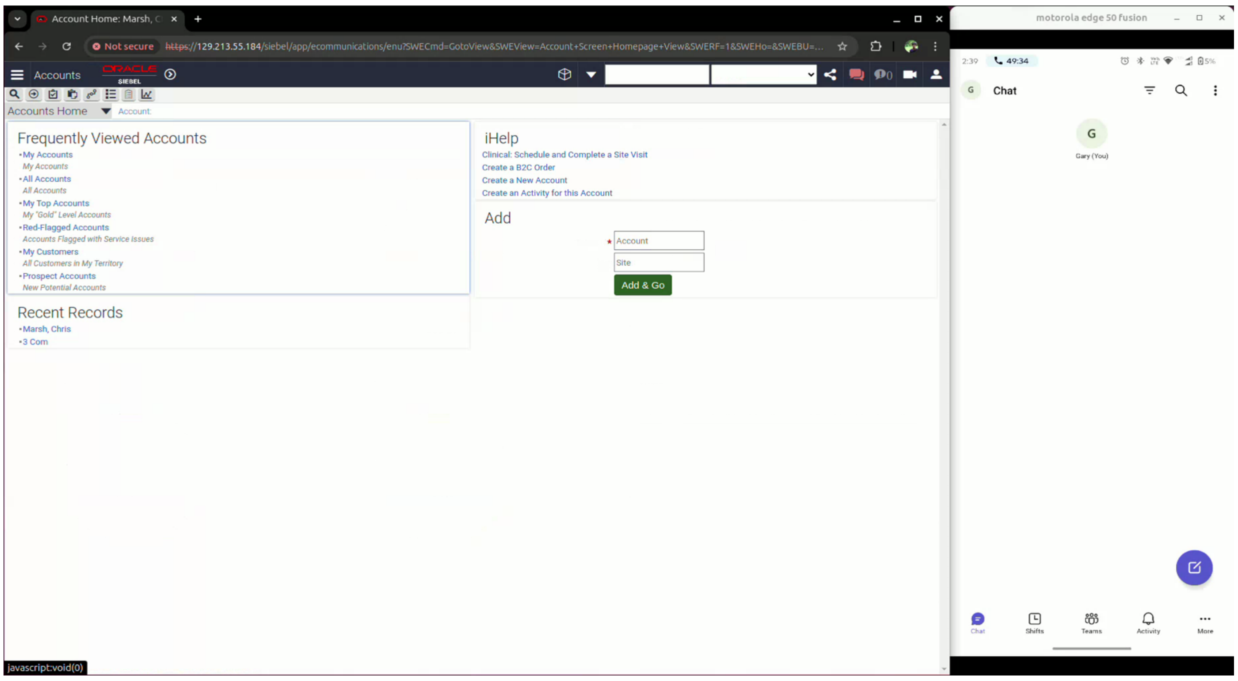
Functionality

Siebel User (Field Service Agent) receives Assigned Appointment details (as a chat message) in their MS Teams application

Steps

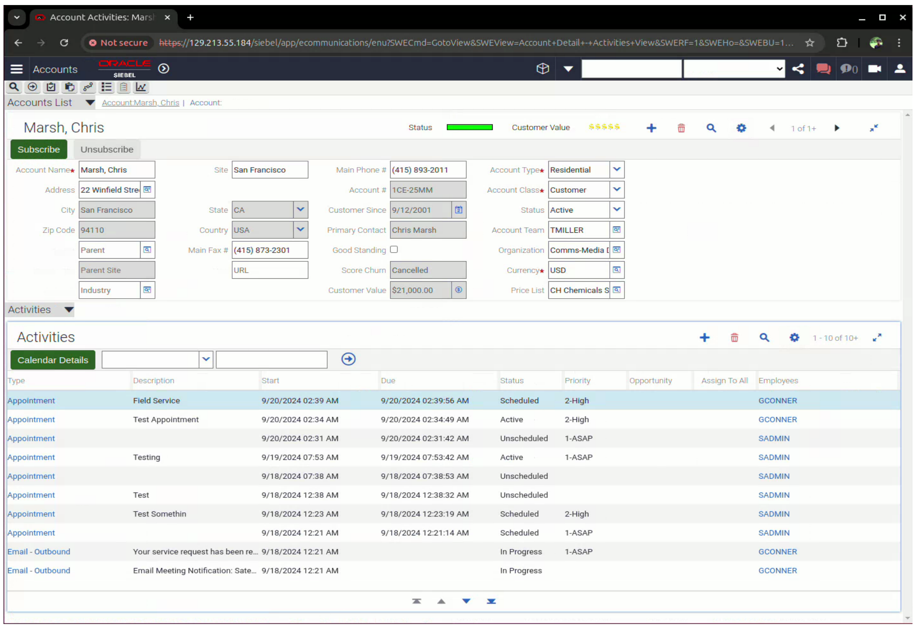
1. Go To Siebel ecommunications UI(Login as any admin user) and go to Accounts Page. Install MS Teams on a mobile device and be logged in to MS Teams as the Field Service Agent.

Figure 1 – Siebel View Logged as Service Administrator & MS Teams App logged in as Field Service Agent



2. Choose the account you want to create the Activity for. From the second applet selector (Drop-down list), choose Activities

Figure 2 - Siebel Account Details Page


Account Activities: Marsh

Not secure https://129.213.55.184/siebel/app/ecommunications/enu?SWECmd=GotoView&SWEView=Account+Detail+-+Activities+View&SWERF=1&SWEHo=&SWEBU=1...

Accounts

Accounts List | Account:Marsh, Chris | Account:

Marsh, Chris — Status — Customer Value $$$$$ — 1 of 1+

Subscribe | Unsubscribe

Account Name*	Marsh, Chris	Site	San Francisco	Main Phone #	(415) 893-2011	Account Type*	Residential
Address	22 Winfield Stre			Account #	1CE-25MM	Account Class*	Customer
City	San Francisco	State	CA	Customer Since	9/12/2001	Status	Active
Zip Code	94110	Country	USA	Primary Contact	Chris Marsh	Account Team	TMILLER
	Parent	Main Fax #	(415) 873-2301	Good Standing	☐	Organization	Comms-Media I
	Parent Site		URL	Score Churn	Cancelled	Currency*	USD
	Industry			Customer Value	$21,000.00	Price List	CH Chemicals S

Activities

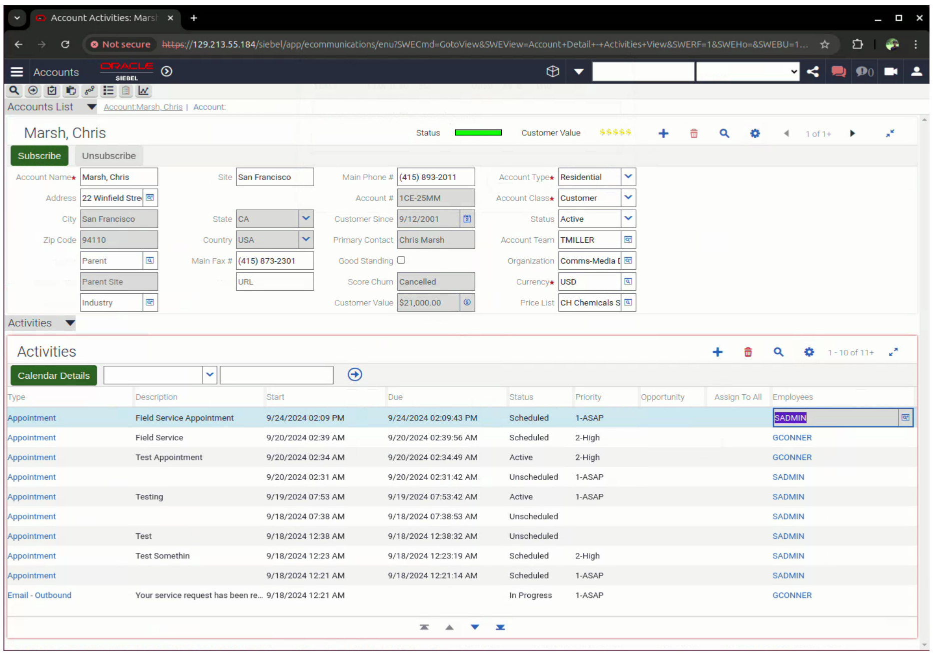
1 - 10 of 10+

Calendar Details

Type	Description	Start	Due	Status	Priority	Opportunity	Assign To All	Employees
Appointment	Field Service	9/20/2024 02:39 AM	9/20/2024 02:39:56 AM	Scheduled	2-High			GCONNER
Appointment	Test Appointment	9/20/2024 02:34 AM	9/20/2024 02:34:49 AM	Active	2-High			GCONNER
Appointment		9/20/2024 02:31 AM	9/20/2024 02:31:42 AM	Unscheduled	1-ASAP			SADMIN
Appointment	Testing	9/19/2024 07:53 AM	9/19/2024 07:53:42 AM	Active	1-ASAP			SADMIN
Appointment		9/18/2024 07:38 AM	9/18/2024 07:38:53 AM	Unscheduled				SADMIN
Appointment	Test	9/18/2024 12:38 AM	9/18/2024 12:38:32 AM	Unscheduled				SADMIN
Appointment	Test Somethin	9/18/2024 12:23 AM	9/18/2024 12:23:19 AM	Scheduled	2-High			SADMIN
Appointment		9/18/2024 12:21 AM	9/18/2024 12:21:14 AM	Scheduled	1-ASAP			SADMIN
Email - Outbound	Your service request has been re...	9/18/2024 12:21 AM		In Progress	1-ASAP			GCONNER
Email - Outbound	Email Meeting Notification: Sate...	9/18/2024 12:21 AM		In Progress				GCONNER

3. Create a new Activity for the Account and add 'Description', 'Status' and 'Priority' for the Activity

Figure 3 - Activity created for the account Marsh, Chris


Account Activities: Marsh

Not secure https://129.213.55.184/siebel/app/ecommunications/enu?SWECmd=GotoView&SWEView=Account+Detail+-+Activities+View&SWERF=1&SWEHo=&SWEBU=1...

Accounts

Accounts List | Account:Marsh, Chris | Account:

Marsh, Chris — Status — Customer Value $$$$$ — 1 of 1+

Subscribe | Unsubscribe

Account Name*	Marsh, Chris	Site	San Francisco	Main Phone #	(415) 893-2011	Account Type*	Residential
Address	22 Winfield Stre			Account #	1CE-25MM	Account Class*	Customer
City	San Francisco	State	CA	Customer Since	9/12/2001	Status	Active
Zip Code	94110	Country	USA	Primary Contact	Chris Marsh	Account Team	TMILLER
	Parent	Main Fax #	(415) 873-2301	Good Standing	☐	Organization	Comms-Media I
	Parent Site		URL	Score Churn	Cancelled	Currency*	USD
	Industry			Customer Value	$21,000.00	Price List	CH Chemicals S

Activities

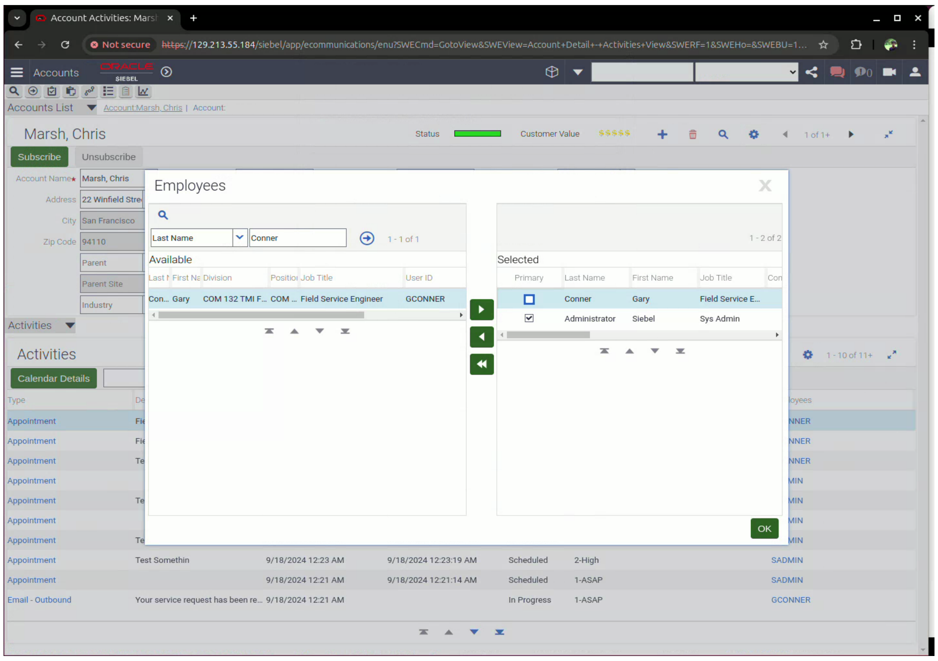
1 - 10 of 11+

Calendar Details

Type	Description	Start	Due	Status	Priority	Opportunity	Assign To All	Employees
Appointment	Field Service Appointment	9/24/2024 02:09 PM	9/24/2024 02:09:43 PM	Scheduled	1-ASAP			SADMIN
Appointment	Field Service	9/20/2024 02:39 AM	9/20/2024 02:39:56 AM	Scheduled	2-High			GCONNER
Appointment	Test Appointment	9/20/2024 02:34 AM	9/20/2024 02:34:49 AM	Active	2-High			GCONNER
Appointment		9/20/2024 02:31 AM	9/20/2024 02:31:42 AM	Unscheduled	1-ASAP			SADMIN
Appointment	Testing	9/19/2024 07:53 AM	9/19/2024 07:53:42 AM	Active	1-ASAP			SADMIN
Appointment		9/18/2024 07:38 AM	9/18/2024 07:38:53 AM	Unscheduled				SADMIN
Appointment	Test	9/18/2024 12:38 AM	9/18/2024 12:38:32 AM	Unscheduled				SADMIN
Appointment	Test Somethin	9/18/2024 12:23 AM	9/18/2024 12:23:19 AM	Scheduled	2-High			SADMIN
Appointment		9/18/2024 12:21 AM	9/18/2024 12:21:14 AM	Scheduled	1-ASAP			SADMIN
Email - Outbound	Your service request has been re...	9/18/2024 12:21 AM		In Progress	1-ASAP			GCONNER

4. Click on the Employee column. From the MVG applet, choose the user corresponding to the MS Teams user and set them as the primary Employee and finally Click on 'Ok'.

Figure 2 - Assign the Activity to Field Service Agent



https://129.213.55.184/siebel/app/ecommunications/enu?SWECmd=GotoView&SWEView=Account+Detail+-+Activities+View&SWERF=1&SWEHo=&SWEBU=1…

Accounts List | Account:Marsh, Chris | Account:

Marsh, Chris

Account Name*	Marsh, Chris
Address	22 Winfield Stre
City	San Francisco
Zip Code	94110

Employees

Last Name: Conner — 1 - 1 of 1

Available

Last N	First Na	Division	Positio	Job Title	User ID
Con...	Gary	COM 132 TMI F...	COM ...	Field Service Engineer	GCONNER

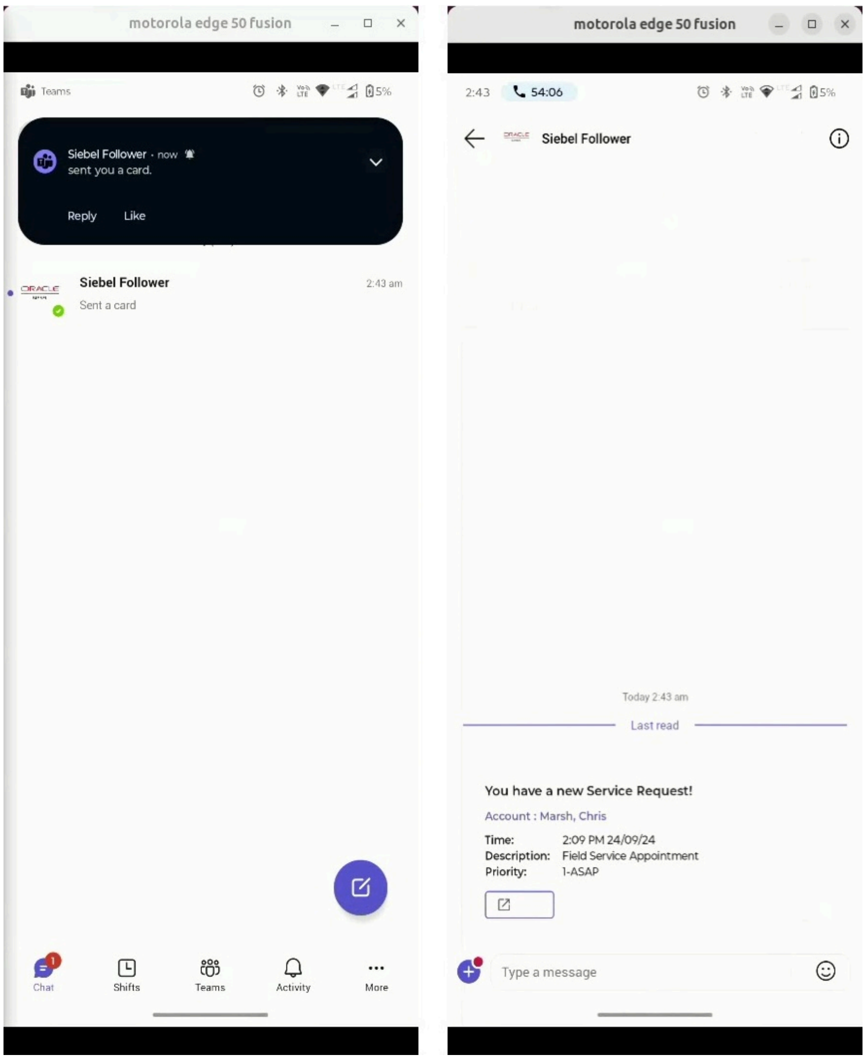
Selected — 1 - 2 of 2

Primary	Last Name	First Name	Job Title	Con
☐	Conner	Gary	Field Service E...	
✓	Administrator	Siebel	Sys Admin	

Activities — 1 - 10 of 11+

Type	De					loyees
Appointment	Fie					NNER
Appointment	Fie					NNER
Appointment	Te					NNER
Appointment						MIN
Appointment	Te					MIN
Appointment						MIN
Appointment	Te					MIN
Appointment	Test Somethin	9/18/2024 12:23 AM	9/18/2024 12:23:19 AM	Scheduled	2-High	SADMIN
Appointment		9/18/2024 12:21 AM	9/18/2024 12:21:14 AM	Scheduled	1-ASAP	SADMIN
Email - Outbound	Your service request has been re...	9/18/2024 12:21 AM		In Progress	1-ASAP	GCONNER

5. Save the record. After a few seconds, you will be notified about the Activity on the MS Teams application. You can click on the deep link attached in the notification message and see the Siebel UI as well



Siebel Follower · now
sent you a card.

Siebel Follower 2:43 am
Sent a card

Siebel Follower

Today 2:43 am

Last read

You have a new Service Request!

Account : Marsh, Chris

Time:	2:09 PM 24/09/24
Description:	Field Service Appointment
Priority:	1-ASAP

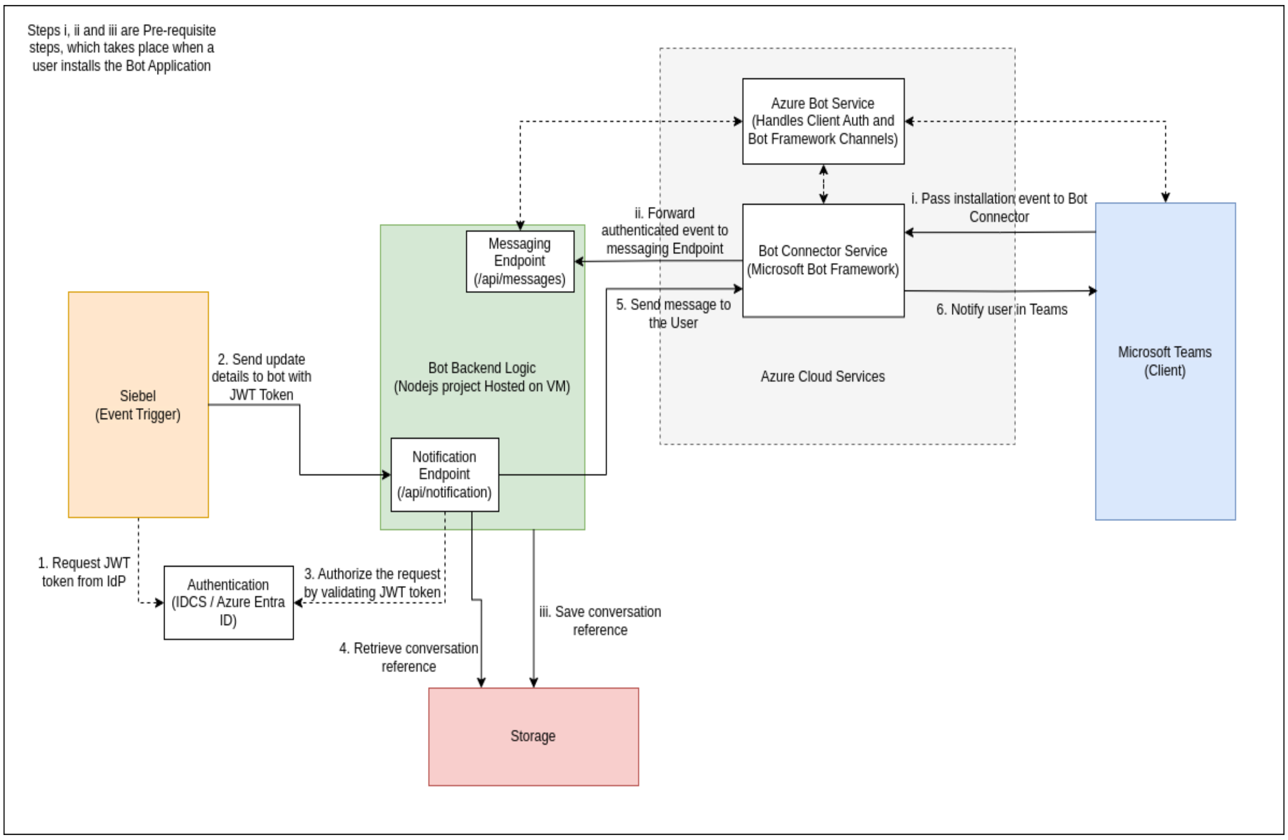
Figure 3 - MS Teams Notification for the Assigned Field Service Agent

Technical Design

MS Teams Custom Bot

MS Teams Custom Bot design will be common for both the User Flows

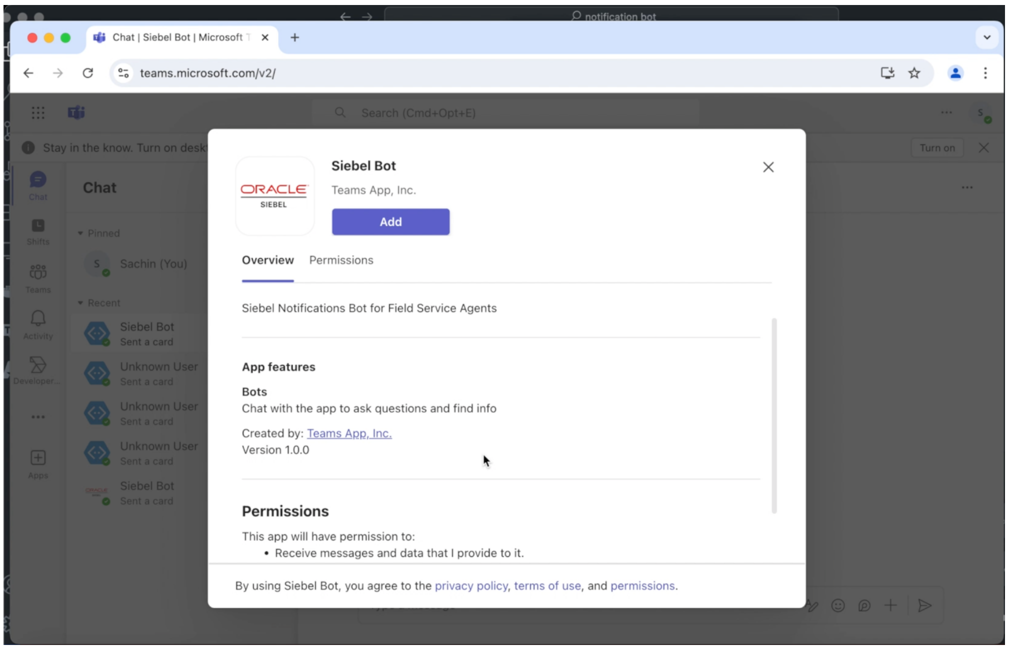
Figure 4 - MS Teams Custom Bot Technical Design


Steps i, ii and iii are Pre-requisite steps, which takes place when a user installs the Bot Application
Azure Bot Service (Handles Client Auth and Bot Framework Channels)
Bot Connector Service (Microsoft Bot Framework)
Azure Cloud Services
i. Pass installation event to Bot Connector
ii. Forward authenticated event to messaging Endpoint
Messaging Endpoint (/api/messages)
Bot Backend Logic (Nodejs project Hosted on VM)
Notification Endpoint (/api/notification)
5. Send message to the User
6. Notify user in Teams
Microsoft Teams (Client)
Siebel (Event Trigger)
2. Send update details to bot with JWT Token
1. Request JWT token from IdP
Authentication (IDCS / Azure Entra ID)
3. Authorize the request by validating JWT token
4. Retrieve conversation reference
iii. Save conversation reference
Storage

Pre-Requisite Steps

1. The user installs the custom bot app in their Microsoft Teams environment. Teams Client creates an event and passes it to the Messaging endpoint(/api/messages), through Bot Connector Service

   Figure 14 - User Adds the Bot on their MS Teams Application



Siebel Bot

Teams App, Inc.

Siebel Notifications Bot for Field Service Agents

App features

Bots

Chat with the app to ask questions and find info

Created by: Teams App, Inc.

Version 1.0.0

Permissions

This app will have permission to:

- Receive messages and data that I provide to it.

By using Siebel Bot, you agree to the privacy policy, terms of use, and permissions.

2. Event arrives at the messaging endpoint as HTTP Request. Bot Framework SDK validates the Authentication Token and the payload in the request

3. Bot Framework SDK processes the Conversation Reference obtained from the event (which includes user details like the user ID) and saves it in storage according to the specifications done in the Backend Logic. As our bot is a Proactive bot, we need to save Conversation Reference to initiate the chat with a user. Bot-Framework-SDK provides in-built methods to find a particular user's conversation reference using their email-id.

Bot Backend Flow

1. On Event trigger, Siebel requests a JWT token from Azure AD by using the client ID and client secret obtained from the respective identity provider (IdP). This token is necessary to access the protected backend logic of the custom bot. (Client Id and Client secret could be stored in Siebel as environment variables and be accessed through code.

2. On obtaining the JWT Token, Siebel sends a POST request to the bot's protected notification endpoint(/api/notifications). This request includes the activity details and the email ID of the assigned user

3. The bot backend logic checks the validity of the JWT token to authorize the Siebel client to access the protected Notification API. If the token is valid, bot proceeds with the request.

4. Bot retrieves the conversation reference stored during the bot installation steps (MS Teams Custom Bot Flow steps 1-3) using the user ID obtained from the POST payload

5. If the user is found in the conversation reference storage, the bot uses the sendAdaptiveCard method to send a message to the user. The bot formulates the reply message using Adaptive Cards, ensuring the message is tailored to the activity details provided by Siebel.

6. Teams Client Obtains the message from the bot and sends notification to the User

7. If the user is not found (i.e., they have not installed the bot), the bot returns an error message, indicating that the user needs to install the bot app in Microsoft Teams

Siebel Design Flows

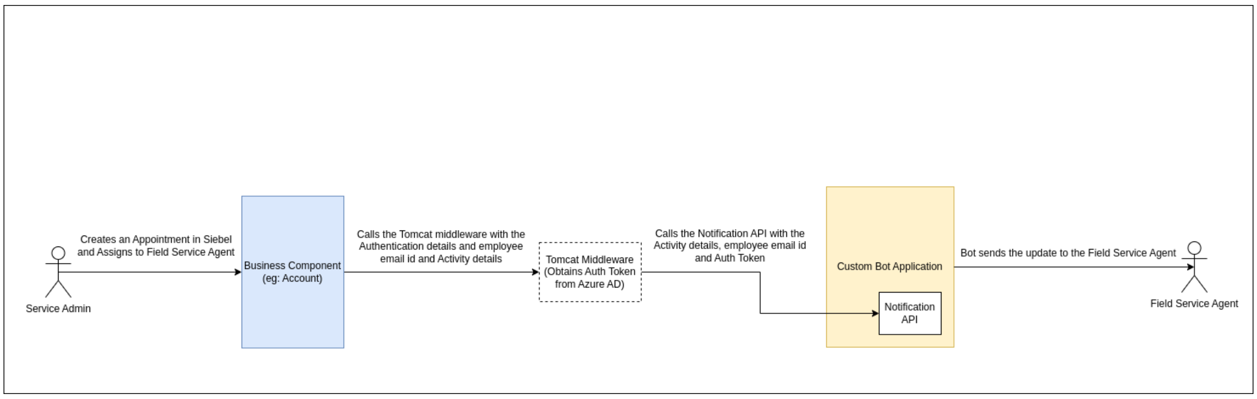

Service Admin
Creates an Appointment in Siebel and Assigns to Field Service Agent
Business Component (eg: Account)
Calls the Tomcat middleware with the Authentication details and employee email id and Activity details
Tomcat Middleware (Obtains Auth Token from Azure AD)
Calls the Notification API with the Activity details, employee email id and Auth Token
Custom Bot Application
Notification API
Bot sends the update to the Field Service Agent
Field Service Agent

Figure 5 - Field Service Agent Use Case - Functional Design

1. 'Service Admin' Creates a new Activity and assigns it to a 'Field Service Agent' in Siebel
2. Upon activity creation, Siebel requests a JWT token from Azure AD by using the client ID and client secret obtained from the respective identity provider (IdP) through a tomcat middleware service
3. On obtaining the JWT Token, Siebel(tomcat middleware) sends a POST request to the bot's protected notification endpoint(/api/notifications). This request includes the activity details and the email ID of the assigned user
4. The bot backend logic checks the validity of the JWT token to authorize the Siebel client to access the protected Notification API. If the token is valid, bot retrieves the user's conversation reference and sends a message to the user by using Adaptive Cards. This message will be tailored to the activity details provided by Siebel

Implementation

MS Teams

We can leverage MS Teams Toolkit to setup a bot which can send notifications to users by using adaptive cards. Teams Toolkit provides boilerplate 'Proactive Notification Bot' code which we can enhance according to our scenario.

Deliverables

We have modified the boilerplate code MS Teams Toolkit provides for creating a Proactive Notification Bot to suit our use case.

FollowBot - compressed.zip

Steps

1. Extract and open the folder using Visual Studio Code.
2. Ensure npm version is 10.8.1 and node version is v20.16.0
3. Install npm packages using *npm install*
4. Run *npm run generate-guid* (generates a unique guid to create the required Azure resources easily through a script)
5. Go to the Teams Toolkit extension and Login into Microsoft 365 account. Run the 'local' script under 'Environment' section in Teams Toolkit extension

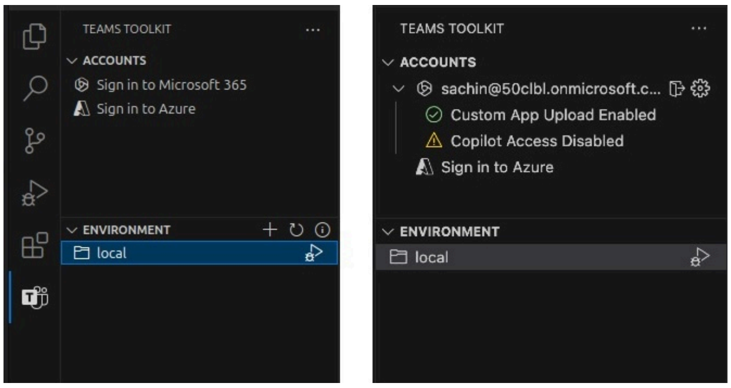


Figure 6 - VS Code Teams Toolkit Extension

6. Login into your teams account in the window that automatically pops up and Install the Bot (Refer: Figure 14)
7. Send a POST request to your Custom Bot for testing

```
{
    "user":"<MS Teams user id (email id)>",
    "type": "Activity",
    "accountName": "3 Com",
    "time": "07/26/2024 03:56:03",
    "url":"https://example.com/",
    "description":"Test Appointment",
    "priority":"1-ASAP",
    "deepLinkUrl": "https://<Siebel deep link url>"
}
```

The code base contains snippets to turn on the Authentication of Notifications API using Azure Entra ID Auth Token.

Run *npm run dev:teamsfx* to run the bot application to run the Custom Bot Application

Security

Notifications invocation Endpoint

We need to make sure that the Notification endpoint that we are exposing from the bot logic to accept Activity details payload allows only authorised access

For authenticating the calls from the Siebel application, we can use either Oracle IDCS/Azure Entra ID as the Identity Provider. We need to verify the jwt token inside the bot backend logic and authorise the access

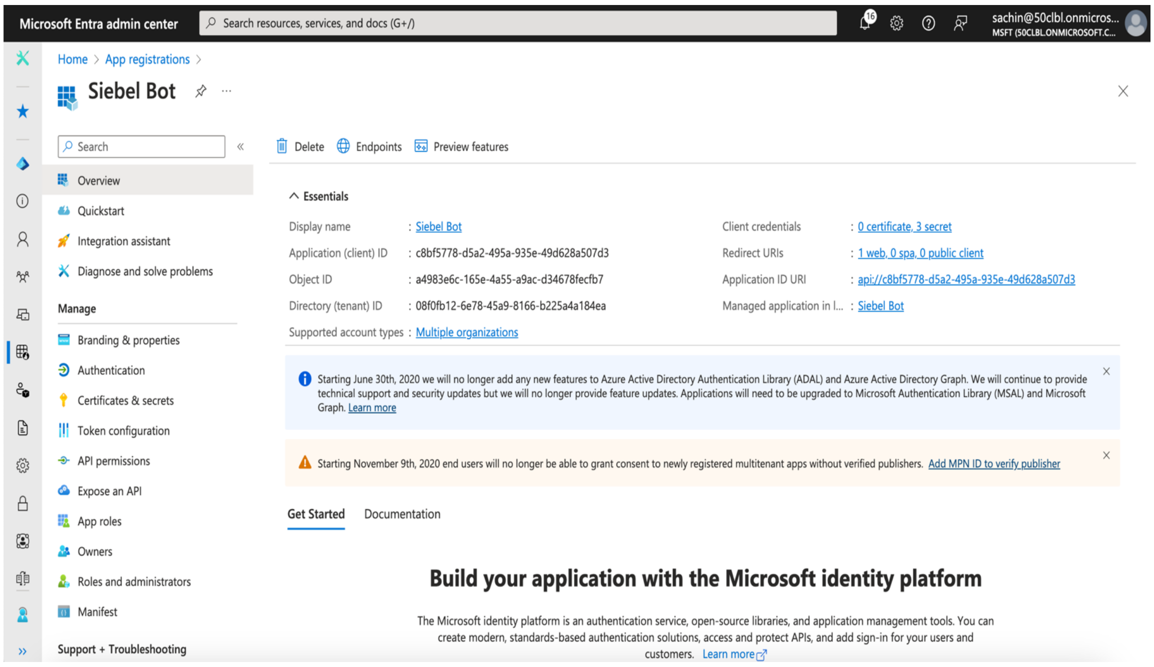
We are exposing 'api/notification' API endpoint from the bot backend server.

Using Azure Entra ID is recommended for securing the Bot Backend logic, as there are npm packages recommended by Azure to verify the Bearer token provided by them.

Azure provide a 'BearerStrategy' by using 'passport' and 'passport-azure-ad' npm packages to verify the Authentication Token provided by them

Configuring a Client Application in Azure AD to access a Resource Application

1. Assuming that we already have an App Registration for our bot app(Automatically gets created after performing the steps in Deliverables section), this would be our Resource Application



Display name	: Siebel Bot
Application (client) ID	: c8bf5778-d5a2-495a-935e-49d628a507d3
Object ID	: a4983e6c-165e-4a55-a9ac-d34678fecfb7
Directory (tenant) ID	: 08f0fb12-6e78-45a9-8166-b225a4a184ea
Supported account types	: Multiple organizations
Client credentials	: 0 certificate, 3 secret
Redirect URIs	: 1 web, 0 spa, 0 public client
Application ID URI	: api://c8bf5778-d5a2-495a-935e-49d628a507d3
Managed application in l...	: Siebel Bot

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

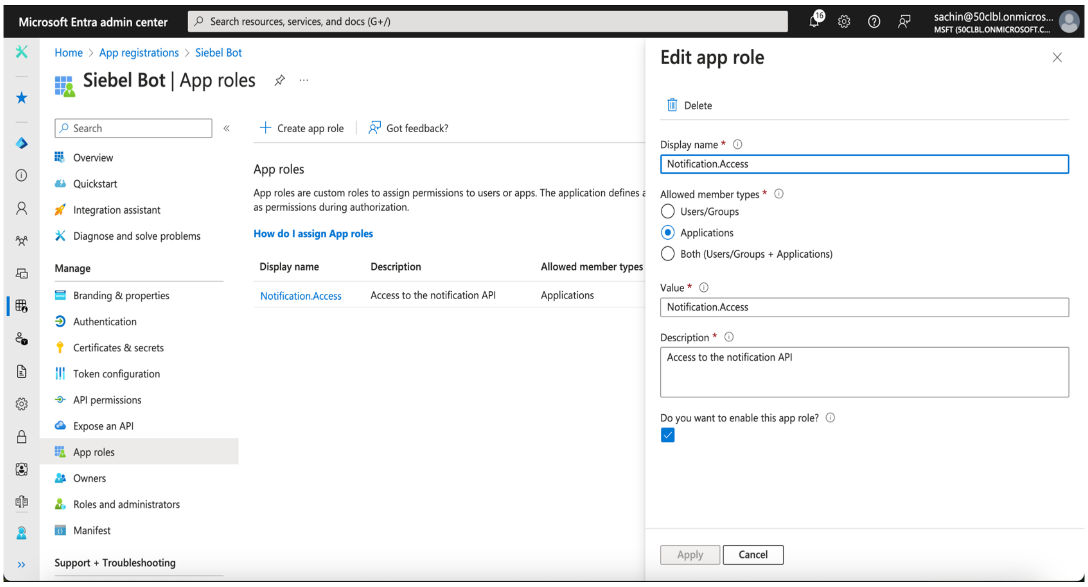
Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multitenant apps without verified publishers. Add MPN ID to verify publisher

Build your application with the Microsoft identity platform

The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern, standards-based authentication solutions, access and protect APIs, and add sign-in for your users and customers. Learn more

Figure 7 - Azure AD App Registration Details Page of the Resource Application (Custom Bot)

2. Click on Application URI in the App Roles tab for the above mentioned App Registration and add a new App Role as below


Microsoft Entra admin center

Home > App registrations > Siebel Bot

Siebel Bot | App roles

App roles

App roles are custom roles to assign permissions to users or apps. The application defines a... as permissions during authorization.

How do I assign App roles

Display name	Description	Allowed member types
Notification.Access	Access to the notification API	Applications

Edit app role

Display name *: Notification.Access

Allowed member types *: ○ Users/Groups [✓ Applications] ○ Both (Users/Groups + Applications)

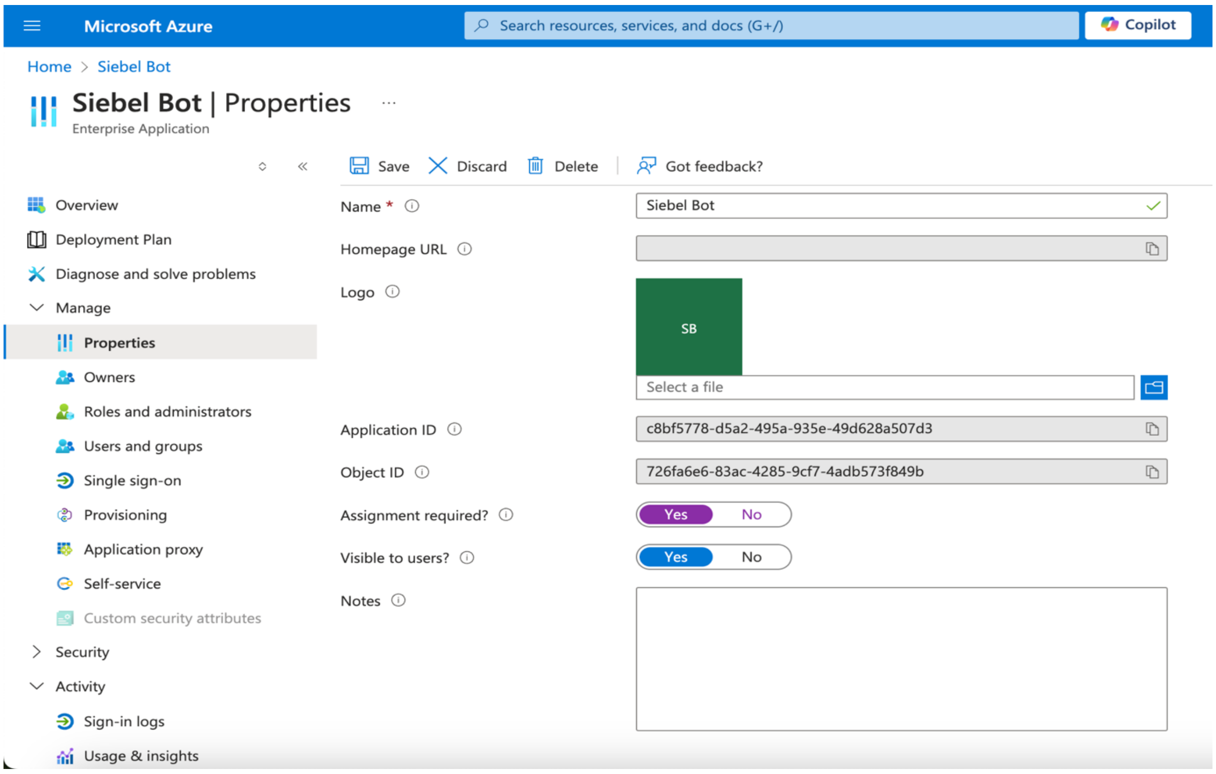
Value *: Notification.Access

Description *: Access to the notification API

Do you want to enable this app role? ✓

Figure 19 - Azure AD App Registrations - Add App Roles for the Resource Application

3. Go to Applications → Enterprise Application in Entra ID and choose the bot. In Manage → Properties tab, make sure 'Assignment required' is turned to 'Yes' - Unless this is turned to 'Yes', every app registration will be able to get an auth token for the Resource application (we'll have to rely on checking for the 'role' claim in the token)


Microsoft Azure

Home > Siebel Bot

Siebel Bot | Properties

Enterprise Application

Name *: Siebel Bot

Homepage URL:

Logo: SB

Application ID: c8bf5778-d5a2-495a-935e-49d628a507d3

Object ID: 726fa6e6-83ac-4285-9cf7-4adb573f849b

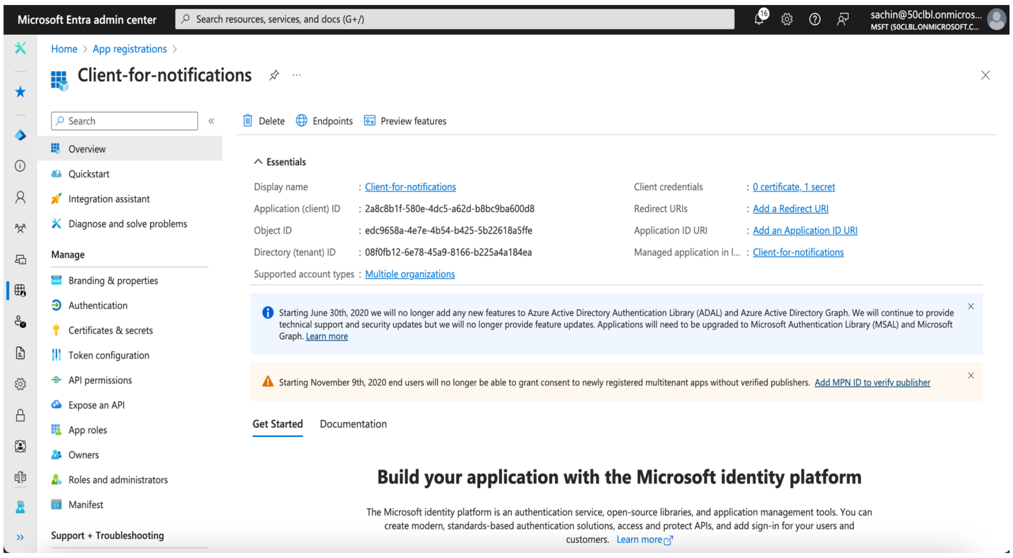
Assignment required?: [✓ Yes] No

Visible to users?: [✓ Yes] No

Notes:

Figure 8 - Azure AD Enterprise Application - Properties View for the Resource Application

4. Create a new App Registration for a Client Application to access the Resource Application



Client-for-notifications

Essentials

Display name	: Client-for-notifications	Client credentials	: 0 certificate, 1 secret
Application (client) ID	: 2a8c8b1f-580e-4dc5-a62d-b8bc9ba600d8	Redirect URIs	: Add a Redirect URI
Object ID	: edc9658a-4e7e-4b54-b425-5b22618a5ffe	Application ID URI	: Add an Application ID URI
Directory (tenant) ID	: 08f0fb12-6e78-45a9-8166-b225a4a184ea	Managed application in l...	: Client-for-notifications
Supported account types	: Multiple organizations		

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

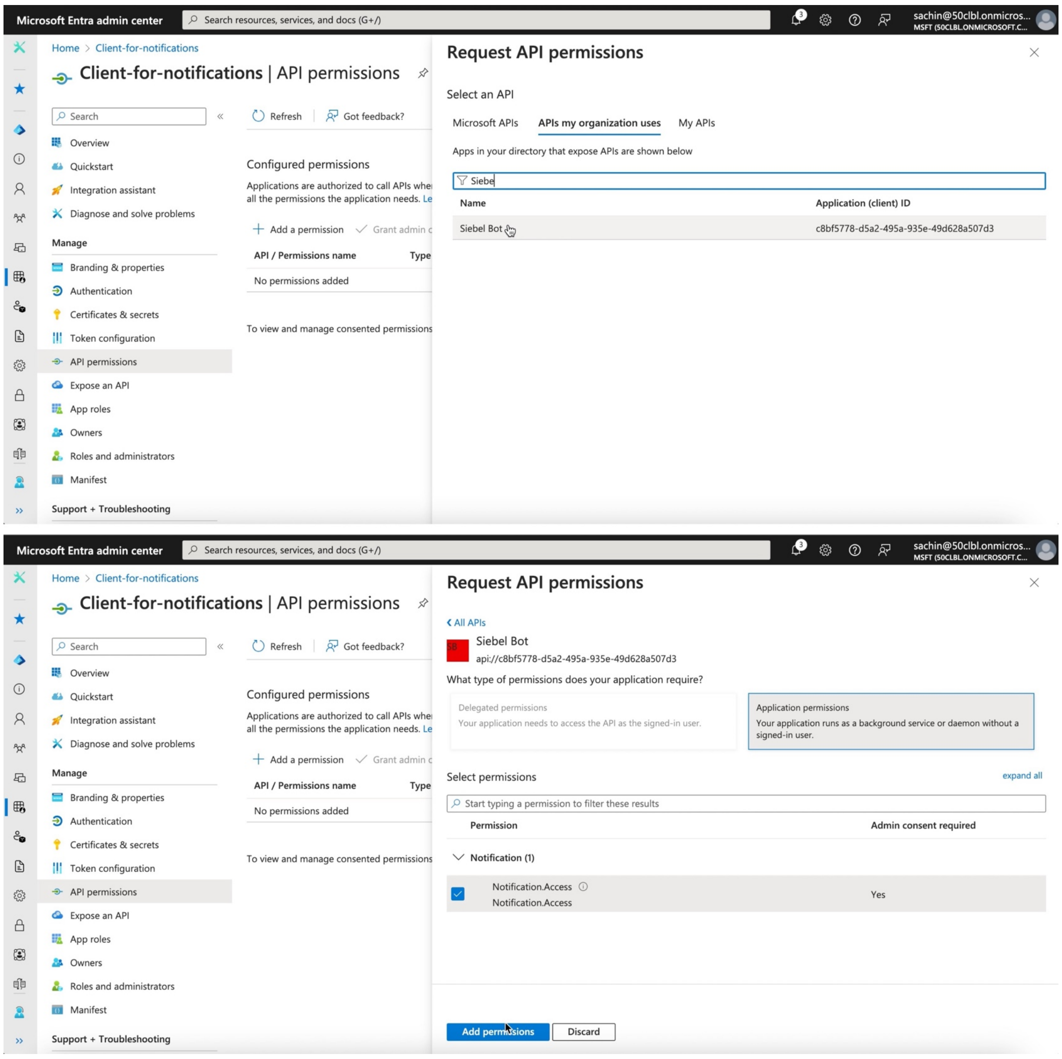
Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multitenant apps without verified publishers. Add MPN ID to verify publisher

Build your application with the Microsoft identity platform

The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern, standards-based authentication solutions, access and protect APIs, and add sign-in for your users and customers. Learn more

Figure 21 - Azure AD App Registrations Page for the Client Application

5. Go to API permissions in the client Applications and add the App Role created in the second step. For this, click on ‘Add a Permission’, Choose ‘APIs my organization uses’, search for the ‘Siebel Bot’ Application and choose ‘Notification.Access’ from Application Permissions



Configured permissions

Applications are authorized to call APIs whe… all the permissions the application needs. Le…

API / Permissions name	Type
No permissions added	

To view and manage consented permissions

Request API permissions

Select an API

Apps in your directory that expose APIs are shown below

Siebe

Name	Application (client) ID
Siebel Bot	c8bf5778-d5a2-495a-935e-49d628a507d3

Request API permissions

Siebel Bot
api://c8bf5778-d5a2-495a-935e-49d628a507d3

What type of permissions does your application require?

Delegated permissions
Your application needs to access the API as the signed-in user.

Application permissions
Your application runs as a background service or daemon without a signed-in user.

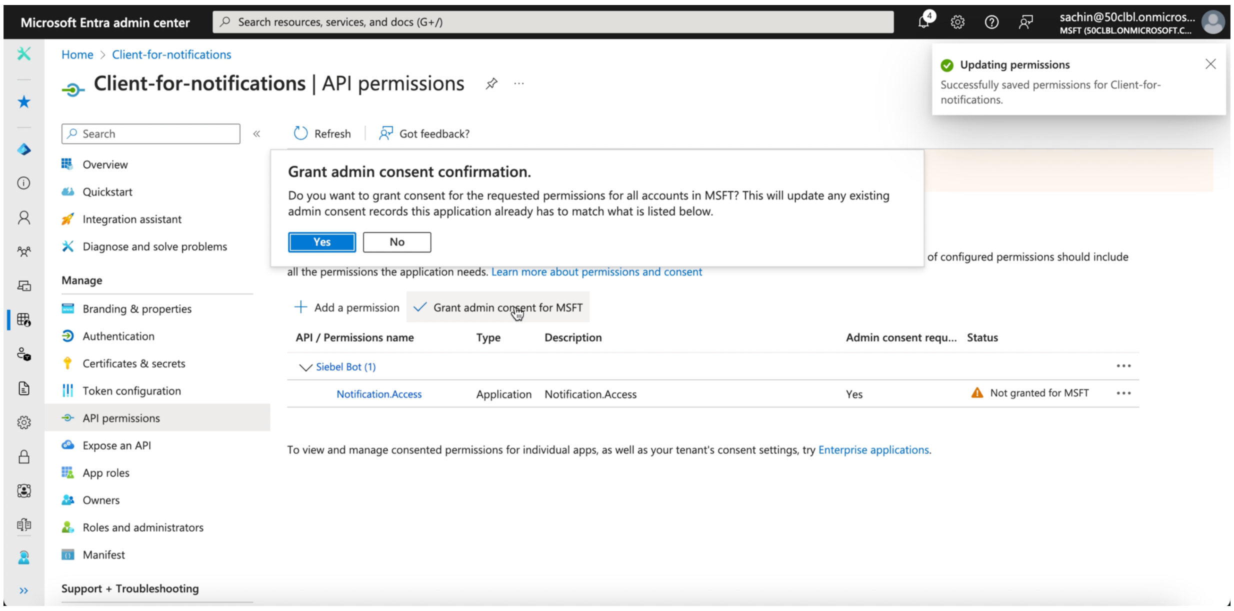
Select permissions

Start typing a permission to filter these results

Permission	Admin consent required
Notification (1)	
✓ Notification.Access Notification.Access	Yes

Figure 22 - Azure AD - Add API Permissions for Client Application

→


Microsoft Entra admin center
Search resources, services, and docs (G+/)
sachin@50clbl.onmicros...
MSFT (50CLBL.ONMICROSOFT.C...
Home > Client-for-notifications
Client-for-notifications | API permissions
Search
Overview
Quickstart
Integration assistant
Diagnose and solve problems
Manage
Branding & properties
Authentication
Certificates & secrets
Token configuration
API permissions
Expose an API
App roles
Owners
Roles and administrators
Manifest
Support + Troubleshooting
Refresh
Got feedback?
Add a permission
Grant admin consent for MSFT
Yes
No

Updating permissions
Successfully saved permissions for Client-for-notifications.

Grant admin consent confirmation.

Do you want to grant consent for the requested permissions for all accounts in MSFT? This will update any existing admin consent records this application already has to match what is listed below.

… of configured permissions should include all the permissions the application needs. Learn more about permissions and consent

API / Permissions name	Type	Description	Admin consent requ...	Status
Siebel Bot (1)				
Notification.Access	Application	Notification.Access	Yes	⚠ Not granted for MSFT

To view and manage consented permissions for individual apps, as well as your tenant's consent settings, try Enterprise applications.

Figure 23 - Azure AD - Grant Admin consent for the added API Permissions

Obtaining token for client credentials flow

```
curl --location 'https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic <base64-encoded<client_id:client_secret>>' \
--data-urlencode 'scope=<scope-for-the-bot>.default'
--data-urlencode 'grant_type=client_credentials' \
```

Authorization of Bearer token from Azure using 'passport' and 'passport-azure-ad' packages in node-js

```
const passport = require('passport');// For Azure Auth for notification endpoint
const BearerStrategy = require('passport-azure-ad').BearerStrategy; // For Azure Auth for
notification endpoint
const { TeamsBot } = require("./teamsBot");

// For Azure Auth for notification endpoint
const options = {
  identityMetadata:
`https://login.microsoftonline.com/${process.env.TEAMS_APP_TENANT_ID}/${config.metadata.vers
ion}/${config.metadata.discovery}`,
  clientID: process.env.BOT_ID,
  audience: process.env.BOT_APP_URI ? process.env.BOT_APP_URI : "api://"+process.env.BOT_ID,
  validateIssuer: config.settings.validateIssuer,
  passReqToCallback: false,
  issuer: `https://sts.windows.net/${process.env.TEAMS_APP_TENANT_ID}/`
}

const bearerStrategy = new BearerStrategy(options, (token, done) => {
  done(null, { }, token);
}
);

const teamsBot = new TeamsBot();

const server = restify.createServer();
server.use(restify.plugins.bodyParser());


server.use(passport.initialize()); // for azure auth for notification endpoint
passport.use(bearerStrategy); // for azure auth for notification endpoint

// POST endpoint protected using Azure token validation
server.post(
  "/api/notification", passport.authenticate('oauth-bearer', {session: false}),
  restify.plugins.queryParser(),
  restify.plugins.bodyParser(), async (req, res) => {
    try {
//Place your logic
        }
    catch(e){
        }
});


server.post("/api/messages", async (req, res) => {
  console.log("Reached")
  await notificationApp.requestHandler(req, res, async (context) => {
    await teamsBot.run(context);
  });
});

server.listen(process.env.port || process.env.PORT || 3978, () => {
  console.log(`\nApp Started, ${server.name} listening to ${server.url}`);
});
```

Messaging Endpoint

When a user installs the Bot in Teams App, these steps take place

- **Teams Client Sends an Event Payload:** Teams client creates the bot installation event and passes it to the Bot connector Service through a channel
- **Connector Receives the Message:** The channel sends the message to the Bot Framework Connector.
- **Connector Forwards Message:** The connector processes the message and forwards it to your bot's web service endpoint.
- **Bot Processes Message:** Your bot receives the message, processes it, and determines a response.
- **Response Sent Back:** The bot sends the response back to the connector, which then forwards it to the channel to be delivered to the user. This could be a welcome message for the user

We have the option to disable the ability to send messages to the bot from Teams Application. But messaging endpoint is still needed to capture Conversation References after bot installation. Security of The Messaging Endpoint and Request Handler of the Bot is handled by the Bot-Framework-SDK internally. Bot-Framework-SDK handles authorization by validating the bearer token provided by the Azure Bot Service.

By design, Request Handler of the Conversation Bot created using Bot-framework SDK handles the requests that are coming to this endpoint. This handler performs the authorization and verifies that the requests being processed are legitimate

- **Connector Service Validation**
  - When the Bot Framework Connector sends requests to your bot, it includes a JWT in the Authorization header.
  - The SDK automatically handles the validation of this token. It checks the token's signature, expiry, issuer, and audience to ensure that the request is coming from a legitimate source.
- **Default Middleware**
  - The Bot Framework SDK comes with built-in middleware that processes incoming requests. This middleware is responsible for extracting and validating the JWT from the Authorization header.
  - If the token is valid, the middleware allows the request to proceed to your bot's logic. If not, the request is rejected.
- **Configuration**
  - Typically, when you create a bot using the Microsoft Bot Framework, you configure it with a Microsoft App ID and a password (Microsoft App Password). These credentials are used by the framework to authenticate and validate requests automatically.
  - This means that if your bot is correctly configured with these credentials, the authentication process is handled seamlessly in the background.

Claims in the Token passed from Teams App to Bot Framework Connector

```
{
  "aud": "https://ic3.teams.office.com",
  "iss": "https://sts.windows.net/08f0fb12-6e78-45a9-8166-b225a4a184ea/",
  "iat": 1723409778,
  "nbf": 1723409778,
  "exp": 1723496478,
  "acct": 0,
  "acr": "1",
  "aio": "AVQAq/8XAAAA......NZciMgHaV+..........+nuKT922AqOob4pz+Y=",
  "amr": [
    "pwd",
    "mfa"
  ],
  "appid": "5e3ce6c0-2b1f-4285-8d4b-75ee78787346",
  "appidacr": "0",
  "given_name": "Sachin",
  "idtyp": "user",
  "ipaddr": "103.....3",
  "name": "Sachin",
  "oid": "da5b38a5-1ede-4253-b26f-abf3ce1ee1ad",
  "puid": "10032003A6588931",
  "rh": "0.AbcAEvvwCHhuqUWBZrIlpKGE6lTwqjmlgcdIpPgCkwEglbn8ALM.",
  "scp": "Teams.AccessAsUser.All",
  "sub": "r3Ayj7edx_qPEfFmkYS76gbeJIfrz2_Hgf-Kwf8vY9c",
  "tid": "08f0fb12-6e78-45a9-8166-b225a4a184ea",
  "unique_name": "sachin@50clbl.onmicrosoft.com",
  "upn": "sachin@50clbl.onmicrosoft.com",
  "uti": "lhSWcOHDqUKVnFgbREFNAA",
  "ver": "1.0",
  "xms_cc": [
    "CP1"
  ],
  "xms_idrel": "12 1",
  "xms_ssm": "1"
}
```

Claims in the Token passed to the bot from the bot framework connector when user sends message to the Bot Expand source

```
{
  "serviceurl": "https://smba.trafficmanager.net/amer/",
  "nbf": 1723411961,
  "exp": 1723415561,
  "iss": "https://api.botframework.com",
  "aud": <bot-id>
}
```

Since we're only interacting with the Bot Framework service and not using any custom APIs or external services that require secure access, then the built-in authentication provided by the Bot Framework SDK should be sufficient.

SSL Certificate and HTTPS Endpoints

Microsoft Bot Framework requires us to provide an endpoint for the messaging endpoint in Bot Framework Portal.

The provided endpoint,

- Should be a HTTPS endpoint
- Should not be an IP address (should have a DNS name)

To have a nodejs restify server serve HTTPS endpoints,

- Obtain a DNS Certificate and the associated private key, either by Obtaining one from a third party or by creating one using
- *openssl req -nodes -new -x509 -keyout server.key -out server.cert*
- Using the obtained certificate and private key, make restify serve https service

```
const fs = require('fs')
const httpsOptions = {key: fs.readFileSync('ssl/key.pem'),
                      certificate: fs.readFileSync('ssl/cert.pem')}
const server = restify.createServer(httpsOptions)
```

If the Virtual Machine running the backend logic does not have a DNS name, we could use reverse proxy services to obtain a DNS name

Storage Considerations

A proactive bot in Microsoft Teams may need to send messages to users even when those users have not directly interacted with the bot recently. To do this, the bot needs to maintain a record of the conversation references. These references allow the bot to locate and re-engage in the previously established conversation or start a new one with the user.

Conversation references contain contextual information about the chat or channel where the bot has interacted with users. This includes:

- **Conversation ID:** Unique identifier for the conversation.
- **User ID:** Identifier for the user.
- **Tenant ID:** Identifier for the tenant.

This contextual information is necessary to send messages or updates to the correct conversation or channel.

Conversation Reference

```
{
  "activityId":"f:4c06e7be-31d2-27d3-2c3f-e2c*****0a",
  "user":{
    "id":"29:xxx",
    "aadObjectId":"xxx"
  },
  "bot":{
    "id":"28:<bot-id>",
    "name":"Siebel Bot"
  },
  "conversation":{
    "conversationType":"personal",
    "tenantId":"<tenant-id>",
    "id":"xxx"
  },
  "channelId":"msteams",
  "locale": "en-GB",
  "serviceUrl":"https://smba.trafficmanager.net/amer/"
}
```

Storage Methods

Azure Storage (Recommended)

Microsoft Bot Framework SDK provides support for keeping the conversation references in Cloud Storage such as Azure Storage.

Cloud Storage is recommended over Local Storage, as local storage does not provide Reliability, Scalability, Security, Backup and Recovery

- Go To Azure Portal. From All Services, choose a new Storage Service
- Create a new storage and create a new Table in the storage
- From Data Storage → Tables, note down the URL(Ignore the '/<table_name>' part at the end) and the table name

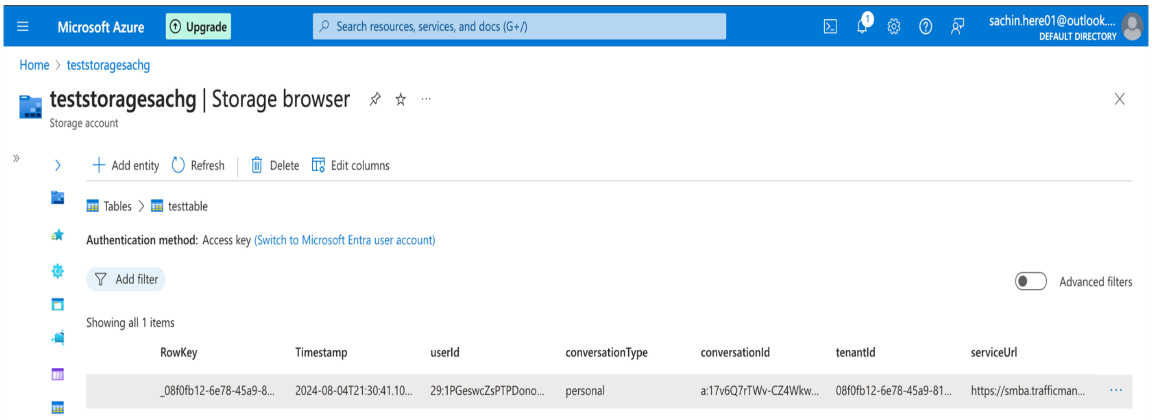
- Choose Security + Networking → Access Keys, Note down the Storage Access key and the Storage name

Implementing the methods to perform actions in Azure Table storage

```
const { AzureNamedKeyCredential, TableClient } = require("@azure/data-tables");
const {
    constructConversationReference,
    extractKeyDataFromConversationReference,
} = require("./util");

class TableStore {
    constructor(storageAccountName, storageAccountURL, storageAccountKey, storageTableName){
        const sharedKeyCredential = new AzureNamedKeyCredential(storageAccountName,
storageAccountKey);

        this.client = new TableClient(`${storageAccountURL}`, `${storageTableName}`,
sharedKeyCredential, { allowInsecureConnection: true });
    }
//add conversation reference logic
    async add(key, reference, options) {
        const task = {partitionKey: this.hash(key), rowKey: key,
...extractKeyDataFromConversationReference(reference),};
        try {
            await this.client.createEntity(task);
            return true;
        } catch (e) {
            return false;
        }
    }
// remove conversation reference logic
    async remove(key, reference {
        try {
            await this.client.deleteEntity(this.hash(key), key);
            return true;
        } catch (e) {
            return false;
        }
    }
//list conversation reference logic
    async list(pageSize, continuationToken) {
        const entities = await this.client.listEntities()
                                 .byPage({ maxPageSize: pageSize, continuationToken:
continuationToken })
                                 .next();

        return {
               data: entities.value.map((entity) => {
                      return constructConversationReference(entity);
                }),
            continuationToken: entities.value.continuationToken,
        };
    }
}

module.exports = TableStore;
```

Initialize the Bot to access Azure Table Storage

```
const { BotBuilderCloudAdapter } = require("@microsoft/teamsfx");
const ConversationBot = BotBuilderCloudAdapter.ConversationBot;
const config = require("./config");
const TableStore = require('../storage') // The above code block

const notificationApp = new ConversationBot({
  adapterConfig: {
    MicrosoftAppId: config.botId,
    MicrosoftAppPassword: config.botPassword,
    MicrosoftAppType: "MultiTenant",
  },
  notification: {
    enabled: true,
       //Table storage configuration
    store: new TableStore("teststoragesachg",
"https://teststoragesachg.table.core.windows.net",
"2lcbfGr*******StorageAccountKey********bA==", "testtable")
  },
});

module.exports = {
  notificationApp
};
```



Authentication method: Access key (Switch to Microsoft Entra user account)

Showing all 1 items

RowKey	Timestamp	userId	conversationType	conversationId	tenantId	serviceUrl
_08f0fb12-6e78-45a9-8...	2024-08-04T21:30:41.10...	29:1PGeswcZsPTPDono...	personal	a:17v6Q7rTWv-CZ4Wkw...	08f0fb12-6e78-45a9-81...	https://smba.trafficman...

Figure 24 - Azure Storage

Local Storage

Local storage is the default method provided by the bot-framework-sdk

Data is stored as an object in,

- *.notification.localstore.json* if running locally.
- *${process.env.TEMP}/.notification.localstore.json*, if *process.env.RUNNING_ON_AZURE* is set to 1.

This method is not recommended for production scenario

Initialize the Bot to access Local Storage

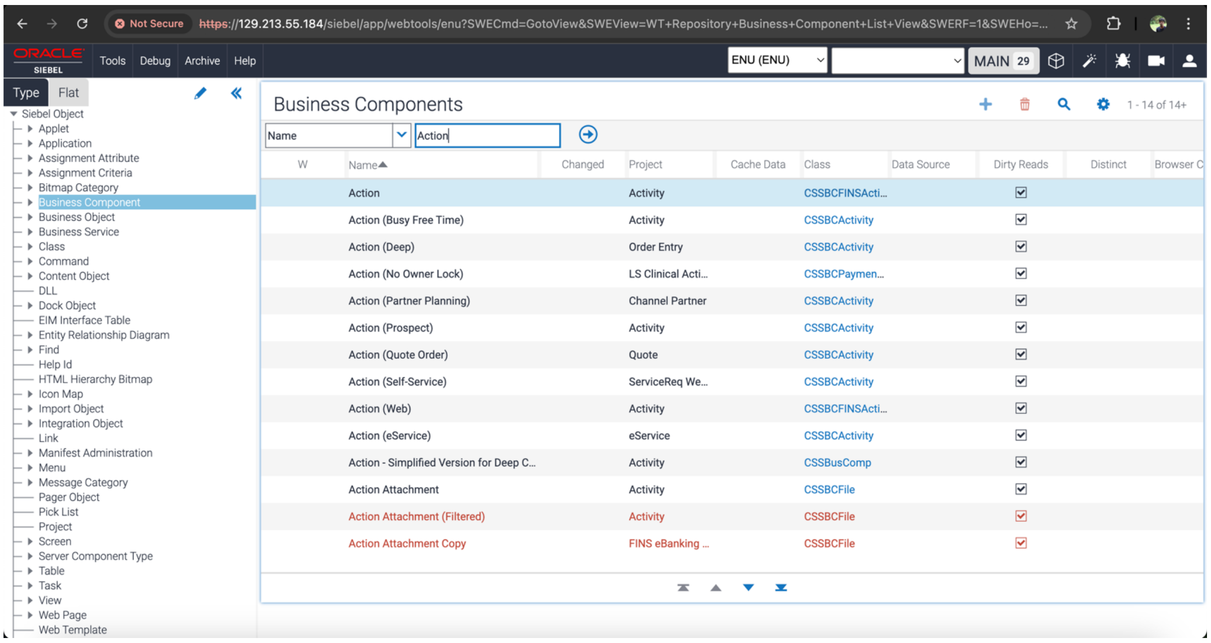
```
const { BotBuilderCloudAdapter } = require("@microsoft/teamsfx");
const ConversationBot = BotBuilderCloudAdapter.ConversationBot;
const config = require("./config");

const notificationApp = new ConversationBot({
  adapterConfig: {
    MicrosoftAppId: config.botId,
    MicrosoftAppPassword: config.botPassword,
    MicrosoftAppType: "MultiTenant",
  },
  // Enable notification
  notification: {
    enabled: true
  },
});
```

```
module.exports = {
  notificationApp,
};
```

Siebel

Setup

We are using Siebel Server Scripts to invoke the Notification API of the Custom Bot. We will be deploying a tomcat service in Siebel to handle the authentication and to act as a middleware



https://129.213.55.184/siebel/app/webtools/enu?SWECmd=GotoView&SWEView=WT+Repository+Business+Component+List+View&SWERF=1&SWEHo=...

Business Components

1 - 14 of 14+

Name: Action

W	Name	Changed	Project	Cache Data	Class	Data Source	Dirty Reads	Distinct	Browser C
	Action		Activity		CSSBCFINSActi...		✓		
	Action (Busy Free Time)		Activity		CSSBCActivity		✓		
	Action (Deep)		Order Entry		CSSBCActivity		✓		
	Action (No Owner Lock)		LS Clinical Acti...		CSSBCPaymen...		✓		
	Action (Partner Planning)		Channel Partner		CSSBCActivity		✓		
	Action (Prospect)		Activity		CSSBCActivity		✓		
	Action (Quote Order)		Quote		CSSBCActivity		✓		
	Action (Self-Service)		ServiceReq We...		CSSBCActivity		✓		
	Action (Web)		Activity		CSSBCFINSActi...		✓		
	Action (eService)		eService		CSSBCActivity		✓		
	Action - Simplified Version for Deep C...		Activity		CSSBusComp		✓		
	Action Attachment		Activity		CSSBCFile		✓		
	Action Attachment (Filtered)		Activity		CSSBCFile		✓		
	Action Attachment Copy		FINS eBanking ...		CSSBCFile		✓		

Figure 9 - Siebel Webtools - Action Business Component

Siebel Server Script code

```
function BusComp_WriteRecord() {
    try {

        var httpService = TheApplication().GetService("EAI HTTP Transport");
        var httpInputs = TheApplication().NewPropertySet();
        var httpOutputs = TheApplication().NewPropertySet();
        var notificationEndpointUrl = "https://<ip-address-of-vm>:3978/api/notification";

        var url = "http://sai-ENT.company.com:4431/SiebelToMSTeamsBot-
1.0/siebelmsteams/eventPusher/pushEvent";
        var descr = this.GetFieldValue("Description");
        var priority = this.GetFieldValue("Priority");

        var result = ContinueOperation;
        this.ActivateField("Employee Email Address")

        var email = this.GetFieldValue("Employee Email Address")
        var accountName = this.GetFieldValue("Account Name");
        this.ActivateField("Start Time")
        var start = this.GetFieldValue("Start Time")
        var duration = this.GetFieldValue("Duration")
        var due = this.GetFieldValue("Due")
        var activityId = this.GetFieldValue("Id")
        var deeplink = [Base-Siebel-Url-for-your-application] +
'?SWECmd=GotoView&SWEView=Activity+List+View&SWERF=1&SWEHo=&SWEBU=1&SWEApplet0=Activity+List
+Applet+With+Navigation&SWERowId0='+ activityId;
```

```
        var msEmail = email;
        var uid = email.split("@")[0];
        if(uid != ""){
          var msEmail = uid + "@50clbl.onmicrosoft.com"; // special handling for changing
the email ids from Siebel to have '50clbl.onmicrosoft.com' email service

          var authUrl = "https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token";
//to obtain auth token from Azure
          var authScope = "<bot-application-id-uri-in-azure>/.default"; //to obtain auth
token from Azure
          var requestBody ='{"userid":"'+ msEmail+
              '","authUrl":"'+authUrl +
              '","authScope":"'+authScope+
              '","url":"'+ notificationEndpointUrl +
              '","message":{"accountName":"'+accountName+
                            '","time":"'+ start +
                            '","description":"'+ descr +
                            '", "deepLinkUrl":"'+ deeplink +
                      '","priority":"'+ priority +'"}}'

          //client-id, client-secret
          var auth = "Basic <base64-encode<client-id:client-secret of client app created in
Azure AD>>";  //to obtain auth token from Azure
          httpInputs.SetProperty("HDR.Authorization", auth);  //to obtain auth token from
Azure
          httpInputs.SetProperty("HTTPRequestURLTemplate", url);
          httpInputs.SetProperty("HTTPRequestMethod", "POST");
          httpInputs.SetProperty("HTTPContentType", "application/json");
          httpInputs.SetProperty("CharSetConversion", "UTF-8");
          httpInputs.SetProperty("HTTPAccept", "*/*");
          httpInputs.SetValue(requestBody);
          httpService.InvokeMethod("SendReceive", httpInputs, httpOutputs);
          }
        } catch (e) {
            TheApplication().SetProfileAttr("Error2", "Error in InvokeREST: " +
e.toString());
            return (CancelOperation);
        }
    }
```

Tomcat Middleware Java Code

Tomcat middleware retrieves the authentication token from the identity provider securing the MS Teams Custom Bot Endpoint and forwards it, along with the incoming payload from Siebel to the Bot Endpoint. This middleware serves as an abstraction layer, simplifying authentication and request handling.

```
package com.siebel.external;
import javax.validation.constraints.Null;
import javax.ws.rs.*;
import javax.ws.rs.client.*;
import javax.ws.rs.core.*;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.node.ObjectNode;
import org.glassfish.jersey.client.ClientConfig;
import org.glassfish.jersey.client.ClientProperties;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

@Path("/eventPusher")
public class SiebelExternalService {

    ObjectMapper mapper = new ObjectMapper();

    @POST
    @Path("/pushEvent")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response pushEvent(String payload, @Context HttpHeaders headers) {

        try {
            JsonNode payloadJson = mapper.readTree(payload);
            String authHeader = headers.getHeaderString("Authorization"); //'Basic
clientId:clientSecret' in base64
            String authScope =
payloadJson.get("authScope")!=null?payloadJson.get("authScope").asText(null) : null;
            String authUrl =
payloadJson.get("authUrl")!=null?payloadJson.get("authUrl").asText(null) : null;

            String accessToken = null;
            if(!Objects.equals(authUrl, null) && !Objects.equals(authUrl, "")&&
!Objects.equals(authScope, null) && !Objects.equals(authScope, "")){
                try {
                    Response idpResponse = obtainToken(authUrl, authHeader, authScope);
                    int statusCode = idpResponse.getStatus();
                    String idpResponseJson = idpResponse.readEntity(String.class);
                    JsonNode idpResponseNode = mapper.readTree(idpResponseJson);
                    if (statusCode != 200) {
                        return Response.status(statusCode).entity(idpResponseJson).build();
                    }
                    accessToken = idpResponseNode.get("access_token").asText();

                } catch (Exception e) {
                    e.printStackTrace();
                    return Response.status(Response.Status.INTERNAL_SERVER_ERROR)
                            .entity("Error obtaining token: " + e.getMessage())
                            .build();
                }
            }


            JsonNode message = payloadJson.get("message");
            String userid =
payloadJson.get("userid")!=null?payloadJson.get("userid").asText(null):null;
            String notificationEndpointUrl =
payloadJson.get("url")!=null?payloadJson.get("url").asText(null):null;
            Map<String, Object> eventPayload = new HashMap<>();
            System.out.println(message);
            if (message != null) {
                parseJsonNode(message, eventPayload);
            }
            eventPayload.put("user", userid);

            MultivaluedMap<String, Object> eventHeaders = new MultivaluedHashMap<>();
            if(accessToken!= null){
                eventHeaders.add("Authorization", "Bearer "+ accessToken);
            }
            eventHeaders.add("Content-Type", "application/json");
            ClientConfig clientConfig = new ClientConfig();
            clientConfig.property(ClientProperties.PROXY_URI, "http://www-
proxy.us.oracle.com:80");
            Client client = ClientBuilder.newClient(clientConfig);
            WebTarget target = client.target(notificationEndpointUrl);
            String jsonEvent = mapper.writeValueAsString(eventPayload);
            Response response = target.request()
                    .headers(eventHeaders)
                    .post(Entity.entity(jsonEvent, "application/json"));

            int statusCode = response.getStatus();
            String responseBody = response.readEntity(String.class);

            if (statusCode == 200 || statusCode == 201) {
                return Response.status(statusCode).entity("message ok").build();
            } else {
                return Response.status(statusCode).entity(responseBody).build();
            }

        } catch (Exception e) {
            e.printStackTrace();
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR)
                    .entity("Error pushing event: " + e.getMessage())
                    .build();
        }
    }
    public void parseJsonNode(JsonNode node, ObjectNode eventPayload) {
        node.fieldNames().forEachRemaining(field -> {
            JsonNode value = node.get(field);

            if (value.isObject()) {
                // If the value is an object, recursively parse it
                parseJsonNode(value, eventPayload);
            } else if (value.isValueNode()) {
                // If it's a simple value, add it to eventPayload
                eventPayload.put(field, value.asText());
            }
        });
    }
    private static Response obtainToken(String authUrl, String authHeader, String authScope)
{

        Form authForm = new Form();
        authForm.param("grant_type","client_credentials");
        authForm.param("scope",authScope);
        ClientConfig clientConfig = new ClientConfig();
        clientConfig.property(ClientProperties.PROXY_URI, "http://www-
proxy.us.oracle.com:80");
        Client client = ClientBuilder.newClient(clientConfig);
        WebTarget target = client.target(authUrl); //works in both Azure and idcs
        Response idpResponse = target.request()
                .header("Authorization", authHeader)
                .post(Entity.entity(authForm, MediaType.APPLICATION_FORM_URLENCODED));

        return  idpResponse;
    }

}
```

References

Design

- Bot Builder Basics
- Bot Activity Handler Concept
- Proactive Installation using Graph

Security

- Configure Authentication in sample node web app with api
- https://github.com/Azure-Samples/ms-identity-javascript-react-tutorial/tree/main/3-Authorization-II/1-call-api
- what-keeps-my-bot-secure-from-clients-impersonating-the-bot-framework-service
- bot-framework-rest-connector-authentication
- https://bitmapbytes.com/idcs-application-oauth2-authentication-with-oracle-content-management/
- https://learn.microsoft.com/en-us/entra/architecture/auth-oauth2

Storage

- https://learn.microsoft.com/en-us/microsoftteams/platform/bots/how-to/conversations/interactive-notification-bot-in-teams
- https://learn.microsoft.com/en-us/javascript/api/@microsoft/teamsfx/botbuildercloudadapter.notificationoptions?view=msteams-client-js-latest
- https://github.com/OfficeDev/teams-toolkit-samples/tree/dev/large-scale-notification

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