Salesforce Chat Integration

## Configuration

Salesforce configuration varies drastically from organization to organization, depending on their requirements and infrastructure. As such, there is no *one* configuration that applies to Maven AGI chat handoff integration.

Handoff integration depends on a Salesforce chat deployment and button that can resolve to an Available state. Any other state will result in the API returning a ChatRequestFailed event.

## Required Properties

* Button ID – The unique identifier for your chat button in Salesforce.
* Chat Host URL – The base URL for your Live Agent server (e.g., <https://d.la12s-core1.sfdc-lywfpd.salesforceliveagent.com>).
* Chat Deployment ID – The unique identifier for your chat deployment.
* ES Live Agent Dev Name – The developer name of the Embedded Service (ES) Live Agent configuration.

### Where to Find These Values

* Navigate to Salesforce Setup > Embedded Service Deployments to find the Deployment ID and ES Live Agent Dev Name.
* Find the Button ID under Chat Buttons & Invitations in Setup.
* The Chat Host URL is typically found in your Salesforce Live Agent settings.

Once you have these details, provide them to our team for integration. If you need assistance, please contact your Salesforce administrator or our support team.

## Examples of Working Configurations in Sandbox

* [Button Id](https://tivoprod--qa.sandbox.my.salesforce-setup.com/lightning/setup/LiveChatButtonSettings/page?address=%2F573O9000001AyGr)
* [Chat Deployment](https://tivoprod--qa.sandbox.my.salesforce-setup.com/lightning/setup/LiveChatDeploymentSettings/page?address=%2F572O90000054FIX)
* [Embedded Services Deployment](https://tivoprod--qa.sandbox.my.salesforce-setup.com/lightning/setup/EmbeddedServiceDeployments/04IO900000F3zLNMAZ/view)
* [Queue](https://tivoprod--qa.sandbox.my.salesforce-setup.com/lightning/setup/Queues/page?address=%2Fp%2Fown%2FQueue%2Fd%3Fid%3D00GO900000EavHR)
* [Site](https://tivoprod--qa.sandbox.my.salesforce-setup.com/lightning/setup/CustomDomain/page?address=%2F0DMO90000005w4z)
* [Omni-Channel Settings](https://tivoprod--qa.sandbox.my.salesforce-setup.com/lightning/setup/OmniChannelSettings/home)
* [Omni-Channel Routing](https://tivoprod--qa.sandbox.my.salesforce-setup.com/lightning/setup/QueueRoutingConfig/page?address=%2F0K90U0000008OPR)

# Deploying the App Via JS Widget

1. Update the Content Security Protocol to allow content from chat.onmaven.app.
2. Launch the `widget.js` script on the host site.

<script src='https://chat.onmaven.app/js/widget.js' defer></script>



1. Initialize the widget and provide configuration settings:

addEventListener("load", function () {

Maven.ChatWidget.load({

organizationId: "tivo",

agentId: "support",

bgColor: "#01a7df",

signedUserData: "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9...",

unsignedUserData: {

todaysDate: Date().toLocaleString()

}

})

});



NOTE: The ID values for the organization and agent are the plain-text versions.

## Configuration Options

interface WidgetConfig {

bgColor?: string; // Widget background color

textColor?: string; // Widget text color (default: 'white')

horizontalPosition?: 'left' | 'right'; // Widget position (default: 'right')

verticalPosition?: 'top' | 'bottom'; // Widget position (default: 'bottom')

organizationId: string; // Required: Your organization ID

agentId: string; // Required: Your agent ID

signedUserData?: string; // See below

unsignedUserData?: Record<string, any> // Arbitrary data to include in conversation for context

}



NOTE: The ID values for the organization and agent are the plain-text versions.

## Encrypting and Signing User Data

When integrating Maven's widget, you'll need to securely transmit user data using a two-step process: signing and encryption. First, configure your app settings by adding your encryption secret and public key during the app installation. Then, implement a server-side function similar to this example:

import { SignJWT, EncryptJWT } from 'jose';

async function secureUserData(userData: Record<string, string> & {

id: string

firstName: string

lastName: string

} & (

{ email: string, phoneNumber?: string } |

{ email?: string, phoneNumber: string }

)) {

// 1. Sign the user data with your private key (ES256 algorithm)

const signedJWT = await new SignJWT(userData)

.setProtectedHeader({ alg: 'ES256' })

.setIssuedAt()

.setExpirationTime('1d')

.sign(yourPrivateKey);

// 2. Encrypt the signed JWT using your encryption secret

const encryptedJWT = await new EncryptJWT({ jwt: signedJWT })

.setProtectedHeader({ alg: 'dir', enc: 'A128CBC-HS256' })

.encrypt(base64url.decode(encryptionSecret));

return encryptedJWT;

}



NOTE: User data must include at least:

* firstName
* lastName
* id
* One of:
  + phoneNumber
  + email