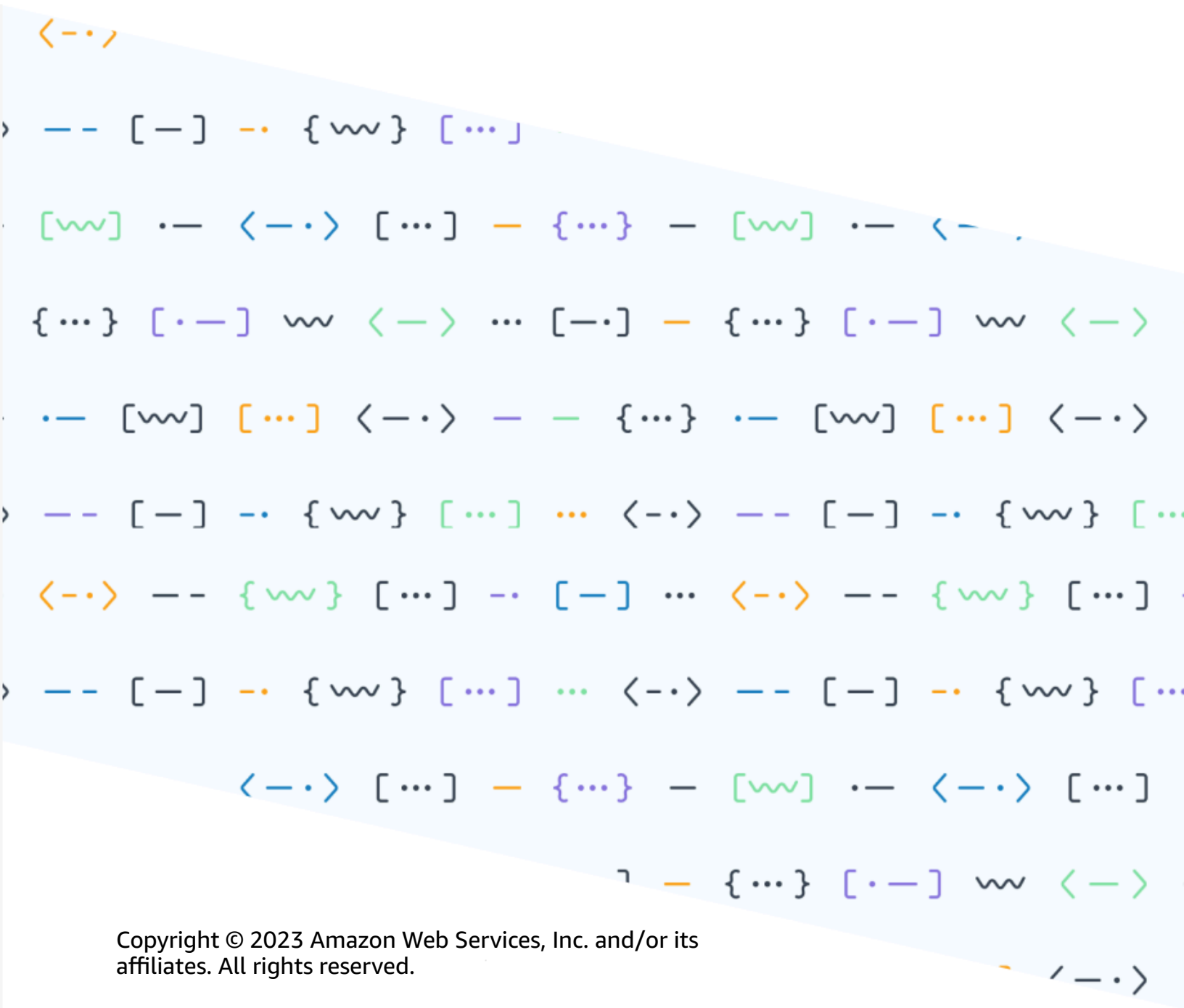
aws

Developer Guide

Agent Workspace



Agent Workspace: Developer Guide

Table of Contents

What is the Amazon Connect Agent Workspace? 1
Are you a first-time Amazon Connect Agent Workspace user? 1
Getting started 2
Prerequisites 2
Creating your application 2
Installing the Amazon Connect SDK 3
Initialize the SDK in your application 3
Testing your application locally 4
Creating an application and associating to your instance 4
Test with a deployed version of your application 6
Integrating with the Amazon Connect Agent Workspace 7
How applications are loaded in the agent workspace 7
Lifecycle events 8
Create 8
Destroy 9
Error handling 9
Recommendations and best practices 10
Ensuring that apps can only be embedded in the Connect agent workspace 10
Using multiple domains within an app 10
Authentication 10
Initializing streams 10
Appendix 11
Role required for creating applications 11
Document history 12

What is the Amazon Connect Agent Workspace?

This is prerelease documentation for a service in preview release. It is subject to change.

Amazon Connect Agent Workspace is a single, intuitive application that provides your agents with all of the tools and step-by-step guidance they need to resolve issues efficiently, improve customer experiences, and onboard faster. Contact center agents might be required to use more than seven applications to manage each customer interaction, digging through various tools to process simple requests, and frustrating customers on hold. Amazon Connect Agent Workspace integrates all of your agent tools on one screen. You can customize the workspace to present agents with step-by-step guidance to resolve customer issues faster.

Are you a first-time Amazon Connect Agent Workspace user?

If you are a first-time user of Amazon Connect Agent Workspace, we recommend that you begin by reading the following sections:

- Customize the Amazon Connect Agent Workspace.
- Third-party applications (3p apps) in the agent workspace (Preview).
- Getting started with third-party development in the Amazon Connect Agent Workspace.
- Integrating with the Amazon Connect Agent Workspace.

Getting started with third-party development in the Amazon Connect Agent Workspace

This is prerelease documentation for a service in preview release. It is subject to change.

You have the option to use first-party applications, such as Customer Profiles, Cases, Wisdom, and features such as step-by-step guides. With support for third-party applications (3p apps), you can unite your contact center software, built by yourself or by partners in one place. For example, you can integrate your proprietary reservation system or a vendor-provided metrics dashboard, into the Amazon Connect agent workspace.

Topics

- Prerequisites
- Creating your application
- Initialize the SDK in your application
- Testing your application locally
- Test with a deployed version of your application

Prerequisites

Developing and testing an application for Connect requires:

- A Connect instance.
- An IAM user that has the proper permissions for creating an application and associating it with the instance. For more information on the required user permissions, see the Appendix.
- A Connect user in that instance that has permissions to update security profiles.

Creating your application

An application is a website that can be loaded from an HTTPS URL into an iframe in the agent workspace. It can be built using any frontend framework and hosted anywhere as long as it can be loaded by the user's browser and supports being embedded. In addition to being accessible by

the user, the application must integrate the application SDK to establish secure communication between the application and the workspace allowing the application to receive events and data from the workspace.

Installing the Amazon Connect SDK

The *Amazon Connect SDK* can be installed from NPM. The SDK is made up of a set of modules that can be installed as separate packages, meaning that you should only pull in the packages that you need. At the moment, there is only the app module, but in the future there will be modules for integrating with contacts and agent events.

The *app* package provides core application features like logging, error handling, secure messaging, and lifecycle events, and must be installed by all applications at a minimum to integrate into the workspace.

Install from NPM

Install the app package from NPM by installing @amazon-connect/app.

```
% npm install --save @amazon-connect/app
```

Initialize the SDK in your application

Initializing the SDK in your app requires calling `init` on the AmazonConnectApp module. This takes an `onCreate` and `onDestroy` callback, which will be invoked once the app has successfully initialized in the workspace and then when the workspace is going to destroy the iframe the app is running in. These are two of the lifecycle events that your app can integrate with. See Lifecycle events for details on the other app lifecycle events that your app can hook into.

```
import { AmazonConnectApp } from "@amazon-connect/app";

const { connectApp } = AmazonConnectApp.init({
  onCreate: (event) => {
    const { appInstanceId } = event.context;
    console.log('App initialized: ', appInstanceId);
  },
```

```
  onDestroy: (event) => {
    console.log('App being destroyed');
  },
});
```

Doing a quick test locally by loading your app directly will produce an error message in the browser dev tools console that the app was unable to establish a connection to the workspace. This will happen when your app is correctly calling `init` when run outside of the workspace.

```
> App failed to connect to workspace in the allotted time
```

Testing your application locally

Once you have a minimal version of the app with the SDK that you want to test in the agent workspace, run your app locally and create an application in the AWS console with an *AccessUrl* using the localhost endpoint, like `http://localhost:3000`.

Creating an application and associating to your instance

> **Note**
>
> Detailed steps for creating and managing applications can be found in the admin guide under Third-party applications (3p apps) in the agent workspace (Preview).

1. Open the Amazon Connect console (https://console.aws.amazon.com/connect/).
2. Navigate to **Third-party applications** in the left hand panel.
3. Choose **Add application**.
4. Fill out the necessary required information.
    a. Name - The name of the application is what will show up to agents in the app launcher in the agent workspace.
    b. Namespace - Namespace must be unique per application and, in the future, allow for applications to support custom events.

   c. AccessUrl - Set to the localhost url for your application.
5. Select the Amazon Connect instance you are testing with to associate the app with that instance.
6. Choose **Add application** to finish creating your app.
7. Log into your test instance as an admin user.
8. Navigate to **Security profiles** and select the `Admin` security profile.
9. Under **Agent applications** find your application and make sure the `View` permission is selected.

   - Open the agent application `/agent-app-v2`
10. Open your app by choosing the app launcher and selecting your application. Your app will be opened in a new application tab.

After following these steps you will have your app loaded from your local machine into the workspace. This will only work when loading the agent workspace on your local machine that has the app running on it. If you want to be able to load your app from any browser / computer, then you must deploy your app somewhere that is internet accessible.

Assuming the logging was included from the code snippet above, you should see the following in the console log of your browser's dev tools when you open your app in the workspace.

```
App initialized:  00420d405e
```

When your app is closed, for example, by closing the tab in the agent workspace, you should see the following series of logs entries.

```
> App destroyed: begin
> App being destroyed
> App destroyed
> App destroyed: end
```

If you see these, then your app correctly integrates with the *Amazon Connect SDK* and the Create / Destroydestroy lifecycle events.

Test with a deployed version of your application

When ready, deploy your app to a place that is internet accessible and update your application configuration (or configure a new application) to point to the deployed version of your application. A simple way to deploy your app assuming it only has static assets is to host them on S3 and (optionally) use Cloudfront.

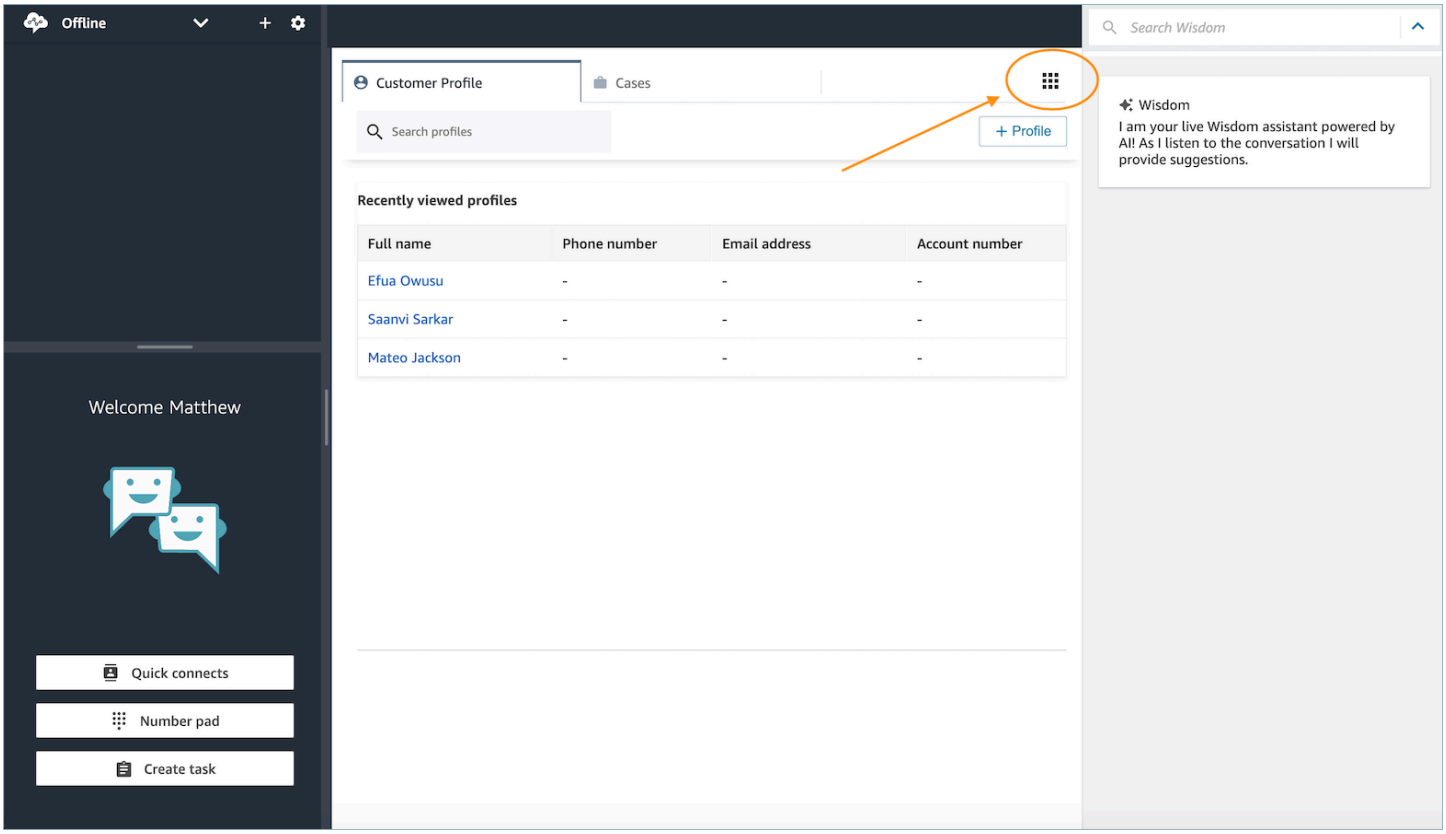
Integrating with the Amazon Connect Agent Workspace

This is prerelease documentation for a service in preview release. It is subject to change.

Applications are loaded into the agent workspace into an iframe, and they won't have access to most things on the parent window because it will be on different domains. Integrating with the agent workspace will require communication between the iframe and the workspace through a MessageChannel for secure inter window communication. Events and messages are supported to provide applications with application lifecycle events, logging, and error handling.

How applications are loaded in the agent workspace

The agent workspace allows users to handle multiple contacts concurrently. They will have only one contact selected at a time though, and the workspace will update the experience based on the channel (call, chat, or task) of the contact and the applications opened for that contact. When a user switches to another contact, the set of application tabs are updated to what the user was doing last when they were on the previous contact.



An application can be opened by the user selecting the app launcher icon in the top right hand corner of the main workspace and select an application from the list. This will load your app in a new application tab for the contact the user has active at that time, or the idle state if the user doesn't have any active contacts. There will be new iframe created for each contact an application is opened with. That iframe will exist until the application tab is closed, for example, a user clicking on the x on the tab or the contact closing. At which point, the app will go through the destroy lifecycle process which gives apps a chance to clean up any resources before the iframe is unmounted from the DOM. The iframe will be hidden when a user selects another tab on the same contact or switches to another contact. This means that at any one time there can be multiple instances, for example, iframes, of the same application running for different contacts.

The agent workspace has a Content Security Policy (CSP) that only allows specific domains to be framed by setting frame-src. The domains configured in the *AccessUrl* and those added to *Approved Origins* will be included in the agent workspace's CSP. Ensure that all domains that your app uses for top level pages are included between *AccessUrl* and *Approved Origins*.

Events and data shared with an instance of an application will be for the contact the application is opened under and the other applications opened on the same contact. Events or data will not be shared between apps on different contacts.

Lifecycle events

There are lifecycle states that an app can move between from when the app is initially opened to when it is closed. This includes the initialization handshake that the app goes through with the workspace after it has loaded to establish the communication channel between the two. There is another handshake between the workspace and the application when the app will be shutdown. An application can hook into `onCreate` and `onDestroy` when calling `AmazonConnectApp.init()`.

Create

The create event results in the `onCreate` handler passed into the `AmazonConnectApp.init()` to be invoked. `Init` should be called in an application once it has successfully loaded and is ready to start handling events from the workspace. The create event provides the *appInstanceId* and the *appConfig*.

- **appInstanceId** - The ID for this instance of the app provided by the workspace.
- **appConfig** - The application configuration being used by the instance for this app.

Destroy

The destroy event will trigger the onDestroy callback configured during AmazonConnectApp.init(). The application should use this event to clean up resources and persist data. The workspace will wait for the application to respond that it has completed clean up for a period of time.

Error handling

Apps can communicate errors back to the workspace by either calling sendError or sendFatalError on the AmazonConnectApp object. The workspace will shutdown an app if it sends a fatal error meaning that the app has reached an unrecoverable state and isn't functional. When an app sends a fatal error the workspace won't attempt to go through the destroy lifecycle handshake and will immediately remove the iframe from the DOM. Apps should do any clean up required prior to sending fatal errors.

Recommendations and best practices

This is prerelease documentation for a service in preview release. It is subject to change.

Ensuring that apps can only be embedded in the Connect agent workspace

It is recommended that apps correctly set the Content Security Policy header with frame-ancestors to only allow Connect instances.

```
Content-Security-Policy: frame-ancestors https://*.awsapps.com https://
*.my.connect.aws;
```

Using multiple domains within an app

Apps that use multiple domains, such as those supporting login flows, must add additional domains to the approved origins list on the application configuration. Both the domain specified in the *AccessUrl* and any additional domains added to the *Approved Origins* will be incorporated into the Content Security Policy for the agent workspace, allowing iframe integration for these domains.

Authentication

Apps must provide their own authentication to their users. It is recommended that apps use the same identity provider that the Connect instance has been configured to use when it was created. This will make it so users only need to log in once for both the agent workspace and their applications, since they both use the same single sign on provider.

Initializing streams

Initializing the CCP via streams, even if hidden, is not supported in third-party applications. You must instead use contact and agent events when they are available.

Appendix

This is prerelease documentation for a service in preview release. It is subject to change.

Role required for creating applications

On top of the `AmazonConnect_FullAccess` IAM policy, users need the following IAM permissions for creating an app and associating it with an Amazon Connect instance.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "app-integrations:CreateApplication",
                "app-integrations:GetApplication",
                "iam:GetRolePolicy",
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "arn:aws:app-integrations:<aws-region>:<aws-account-
Id>:application/*",
            "Effect": "Allow"
        }
    ]
}
```

Document history for the Agent Workspace Developer Guide

The following table describes the documentation releases for Agent Workspace.

Change	Description	Date
Initial release	Initial release of the Agent Workspace Developer Guide	October 27, 2023